# GPIB (IEEE-488) İLE BİR ELEKTRONİK CİHAZIN UZAKTAN KONTROLÜ

**Kenan KUZULUGİL**

**Y.Lisans Tezi**
**Elektrik ve Elektronik Mühendisliği Anabilim Dalı**
**Yrd. Doç. Dr. Tevhit KARACALI**
**2011**

**ATATÜRK ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**Y. LİSANS TEZİ**

# GPIB (IEEE-488) İLE BİR ELEKTRONİK CİHAZIN UZAKTAN KONTROLÜ

**Kenan KUZULUGİL**

**ELEKTRİK VE ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**ERZURUM**
**2011**

T.C.
ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

# TEZ ONAY FORMU

GPIB (IEEE-488) İLE BİR ELEKTRONİK CİHAZIN UZAKTAN KONTROLÜ

Yrd. Doç. Dr. Tevhit KARACALI danışmanlığında, Kenan KUZULUGİL tarafından hazırlanan bu çalışma 25/11/2011 tarihinde aşağıdaki jüri tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda Yüksek Lisan tezi olarak **oybirliği** ile kabul edilmiştir.

Başkan : Doç. Dr. Bülent ÇAKMAK *İmza* :

Üye : Yrd. Doç. Dr. Deniz DAL *İmza* :

Üye : Yrd. Doç. Dr. Tevhit KARACALI *İmza* :

Üye : ..... *İmza* :

Üye : ..... *İmza* :

(imza)
Yukarıdaki sonucu onaylıyorum
**Enstitü Müdürü**

**Not:** Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

# ÖZET

Y.Lisans Tezi

GPIB (IEEE-488) İLE BİR ELEKTRONİK CİHAZIN UZAKTAN KONTROLÜ

Kenan KUZULUGİL

Atatürk Üniversitesi
Mühendislik Fakültesi
Elektrik ve Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Tevhit KARACALI

İlerleyen teknoloji kullanıcıya bir takım avantajların yanı sıra dezavantajlar da sunar. Bu dezavantajların nedeni çoğunlukla kullanılan elektronik cihaz ve malzemelerin senkronize bir şekilde hızla ilerleyen teknolojiye uyarlanamamasıdır. Bu sebeple asli işlevini yerine getirebilen ancak bilgisayarların sahip olduğu teknolojiye sahip olmayan (örneğin USB port gibi) elektronik cihazların kullanımı gittikçe azalıyor. Bu çalışmanın amacı yapılan bir kullanıcı arayüzüyle bilgisayardan GPIB portuna sahip olup USB portuna sahip olmayan bir cihazın uzaktan kontrol edilmesidir. Bu kontrolde bilgisayar ile cihaz arasında USB-GPIB dönüştürücü kullanılmıştır. Bu çalışma için kullanıcı arayüzü için esnek ve kullanıcıya diğer görsel programlama dillerinden daha çok imkan sunan Microsoft Visual C# 2010 programı seçilmiştir. Bu program vasıtasıyla tasarlanan bir kullanıcı arayüzüyle, optik spektrum ölçüm ve analizinde kullanılan MS9710B cihazı, USB-GPIB dönüştürücüsü aracılığıyla uzaktan kontrol edilmiştir. Sonuç olarak, normalde ölçme ve testte yeterli olan bir cihazın yapılan çalışmayla atıl duruma düşmesi engellenerek mevcut teknolojiye ayak uydurması sağlanmıştır.

**2011, 131 sayfa**

**Anahtar Kelimeler:** GPIB, USB-GPIB, Visual C#, Uzaktan Kontrol, Grafik, MS9710B

## ABSTRACT

MS Thesis

REMOTE CONTROL OF AN ELECTRONIC DEVICE BY GPIB (IEEE-488)

Kenan KUZULUGİL

Atatürk University
Faculty of Engineering
Department of Electrical and Electronic Engineering

Supervisor: Asst. Prof. Dr. Tevhit KARACALI

Advanced technology presents advantages as well as disadvantages. The reason of these disadvantages is unable to adapt electronic devices currently used to the rapidly developing technology synchronically. Therefore, the use of electronic devices which can perform essential tasks but do not have the technology the computers have (such as USP port) is getting less. The purpose of this study is to control remotely a device which does not have a USB port but a GPIB port via an interface made on a computer. In this control, a USB to GPIB converter was used. Microsoft Visual C# 2010 which is flexible for user interface and provides more user facility than other visual programming languages was preferred for the study. With a user interface made on this program, the MS9710B device used for optical spectrum measurement and analysis was remote controlled via an USB-GPIB converter. As a result, through this study, it was prevented that a device normally capable of measuring and testing be out of service by adapting it to the current technologies.

**2011, 131 sayfa**



**Keywords:** GPIB, USB-GPIB, Visual C#, Remote Control, Graphics, MS9710B

## TEŞEKKÜR

Yapmış olduğum bu çalışmada öncelikle her türlü destek ve ilgisini esirgemeyen değerli hocalarım Sayın Yrd. Doç. Dr. Tevhit KARACALI ve Sayın Doç. Dr. Bülent ÇAKMAK'a teşekkür ederim.

Tezimin gerçekleştirilmesi aşamasında manevi desteğinden dolayı Gümüşhane Üniversitesi Kelkit Aydın Doğan Meslek Yüksekokulu Müdürü Sayın Doç. Dr. Vecihi AKSAKAL'a ve okulumuz akademik personeline teşekkürü bir borç bilirim.

Ayrıca bu süreçte daimi yanımda olan ve tezimin gerçekleştirilmesi için elinden gelen yardım ve desteklerini esirgemeyen sevgili eşim ve kıymetli aileme sonsuz teşekkür ediyorum.

Kenan KUZULUGİL
Eylül 2011

## İÇİNDEKİLER

## SİMGELER ve KISALTMALAR DİZİNİ

| | |
|---|---|
| A/D | Analog/Dijital |
| API | Application Programming Interface (Uygulama Programlama Arayüzü) |
| ATN | Attention (Dikkat) |
| CIC | Controller in Charge (Görevdeki Kontrolör) |
| DAV | Data Valid (Veri Geçerli) |
| DVM | Digital Voltmeter (Dijital Voltmetre) |
| EOI | End or Identify (Bitir veya Tanımla) |
| GDI | Graphics Device Interface (Grafik Cihaz Arayüzü) |
| GPIB | General Purpose Interface Bus (Genel Amaçlı Arayüz Yolu) |
| GUI | Graphical User Interface (Grafiksel Kullanıcı Arayüzü) |
| HP | Hewlett Packard |
| IBM | International Business Machines |
| IEEE | Institute of Electrical and Electronics Engineering (Elektrik Elektronik Mühendisliği Enstitüsü) |
| IFC | Interface Clear (Arayüz Temizleme) |
| NDAC | Not Data Accepted (Veri Kabul Edilmedi) |
| NRFD | Not Ready For Data (Veri İçin Hazır Değil) |
| PIC | Peripheral Interface Controller (Çevresel Arabirim Denetleyicisi) |
| REN | Remote Enable (Uzak Etkin) |
| SCPI | Standart Commands of Programmable Instruments (Programlanabilir Cihazlar için Standart Komutlar) |
| SRQ | Service Request (Servis İsteği) |
| TTL | Transistor-Transistor Logic |
| USB | Universal Serial Bus (Evrensel Seri Yol) |

# ŞEKİLLER DİZİNİ

# ÇİZELGELER DİZİNİ

# 1. GİRİŞ

Bilimsel çalışmalar teorikte ilerleme kaydetse de bunların doğruluk, güvenilirlik ve gerçeklik payının ortaya konulması için gerçek hayata geçirilmeleri gerekir. Bu başlangıç olarak çok mümkün olmasa da laboratuvar ortamlarında yapılan test, ölçüm, analiz ve deneyler sonucu elde edilen veriler bunun ilk adımıdır. Bu nedenle bu verilerin elde edilmesinde kullanılan bilimsel cihazlar ve verinin işlenmesinde kullanılan bilgisayarlar arasında iletişim kurmak bilimsel çalışma yapanlar için zaman kazandırmanın yanında veri kaybını da aza indirecektir.

## 1.1. Çalışmanın Amacı ve Kapsamı

Yapılan bu çalışmada amaç; kullanılan test ve ölçüm cihazlarının özelliklerini kullanarak günümüzdeki teknolojiye uyarlanmasını sağlamaktır. Optik spektrum ölçüm ve analizinde kullanılan Anritsu firmasının MS9710B optik spektrum analizörü test, ölçüm ve analizde ihtiyacı karşılamasına rağmen elde edilen verileri bilgisayara aktarmada bugünkü teknolojiye göre bir adım geride kalmaktadır. Ölçüm sonuçları yalnızca diskete (floppy diske) kaydedilebilmektedir. Bilindiği üzere günümüzdeki masaüstü bilgisayar, dizüstü ve netbook gibi bilgisayarlarda floppy disk bulunmamaktadır. Bu sebeple cihaz atıl duruma düşmektedir. Bu sorunun ortadan kaldırılması için bu çalışma gerçekleştirilmiştir. Çalışmada cihazın uzaktan kontrolü için GPIB kullanılmasının sebebi birden çok test veya ölçüm cihazının bu sisteme dahil edilebilmesidir.

USB portu günümüz bilgisayar teknolojisinde bilgisayar ve çevresel cihazlar arasında iletişim kuran ve her geçen gün kullanımı yaygınlaşarak artan iletişim birimidir. GPIB portu çoğu bilimsel cihazlarda standart iletişim birimi olmasına rağmen bilgisayarlarda standart olmayan ve bilgisayara sonradan eklenebilen bir ara kart aracılığı ile sağlanmaktadır. Ancak bu kartların bilgisayara takılacak yuvaları günümüz bilgisayarlarında giderek kullanılmamaya başlanmıştır. Bu nedenle USB portunu

kullanmak günümüz ve yakın gelecekte kaçınılmaz olmuştur. Bununla birlikte kendi alanında çok başarılı ölçüm veya analiz yapan bilimsel cihazlar bünyesinde bulunan veri depolama ya da aktarma birimleri çok hızlı değişen bilgisayar teknolojisine göre eski kalmaktadır. Bu dezavantajı yüzünden atıl hale gelen çok pahalı ve değerli olan bilimsel cihazları güncellemek önem kazanmaktadır.

Bu çalışmada, Microsoft firmasının Visual C# programında yapılan bir kullanıcı arayüzü ve Keithley firmasının USB-GPIB dönüştürücüsü ile Anritsu firmasının MS9710B optik spektrum analizörü bilgisayarla uzaktan kontrol edilmiştir. USB-GPIB dönüştürücü sayesinde cihazın GPIB portu ve bilgisayarın USB portu kullanılmıştır. Böylece bilgisayarda GPIB portu olmamasına rağmen USB yolundan cihazın GPIB portuyla haberleşmesi sağlanmıştır. Yapılan arayüz sayesinde ise; cihaza istenilen komutların gönderilmesi, istenilen zamanda cihaza ölçüm yaptırılması, elde edilen sonuçların cihazdan alınması, bu sonuçların arayüz ekranında çizdirilmesi ve çizdirilen bu grafiğin text ve/veya bitmap formatında bilgisayara kaydedilmesi işlemleri gerçekleştirilmiştir.

### 1.2. Çalışmanın Anahtarları

Bu çalışma; giriş, kuramsal temeller, materyal ve yöntem, araştırma bulguları ve tartışma ve sonuç olmak üzere beş ana bölümden oluşmaktadır.

Bölüm 2'de; USB-GPIB dönüştürücüde kullanılan USB ve GPIB protokolleri ile kullanıcı arayüzünün hazırlandığı Visual C# 2010 programı hakkında bilgi verilmektedir.

Bölüm 3'te; Visual C# programındaki grafik sınıfı hakkında, kullanıcının cihaz ile haberleşmesi için yapılan programda kullanılan USB-GPIB dönüştürücüsünün C#'daki komut ve yordamları ile Anritsu firmasının MS9710B optik spektrum analizörü hakkında ayrıntılı bilgi verilmektedir.

Bölüm 4'te, kullanıcının cihazı uzaktan kontrol etmesi için yapılan kullanıcı arayüzü ve bu arayüzün kullanımı ve özellikleri hakkında bilgi verilmektedir.

Bölüm 5'te, yapılan çalışmayla neler gerçekleştirilebildiği ve bu tezle hangi sorunun üstesinden gelindiği hakkında bilgi verilmektedir.

# 2. KURAMSAL TEMELLER

## 2.1. USB

### 2.1.1. Tanım

USB, Evrensel Seri Yol, bilgisayarlara çeşitli çevre birimlerin takılabilmesi için geliştirilmiş bir veri yolu standardıdır. Bilgisayar endüstrisinin ileri gelen firmalarından Compaq, HP, IBM, Intel ve Microsoft firmaları tarafından Kasım 1994'te 0,7 sürümü ile başlayan süreç, Kasım 2008'de 3,0 sürümü ile devam etmektedir. USB'nin kısa tarihçesi Çizelge 2.1'de özet olarak gösterilmiştir (Hewlett Packard 2011).

**Çizelge 2.1.** USB kronolojisi

| Sürüm | Tarih |
|---|---|
| 0,7 | 11 Kasım 1994 |
| 0,8 | 30 Aralık 1994 |
| 0,9 | 13 Nisan 1995 |
| 0,99 | 25 Ağustos 1995 |
| 1,0 FDR | 13 Kasım 1995 |
| 1,0 | 15 Ocak 1996 |
| 1,1 | 23 Eylül 1998 |
| 2,0 (0.79 tasarısı) | 5 Ekim 1999 |
| 2,0 (0.9 tasarısı) | 21 Aralık 1999 |
| 2,0 | 27 Nisan 2000 |
| 3,0 | 12 Kasım 2008 |

USB'nin kullanım alanlarının başında ölçme ve kontrol gelmektedir. USB, kullanıcıya ve tasarımcıya sağladığı avantajları ile günümüz teknolojisinin vazgeçilmez veri yolu standartlarından biri olmuştur. USB tasarımının 127 adet çevre birimin bağlanabilmesine izin vermesi ve diğer avantajları göz önüne alındığında bu proje için en uygun veri yolu standardı olarak karşımıza çıkmıştır. USB'nin kullanıcıya sağladığı avantajlar şunlardır:

• **Kullanım kolaylığı:** USB'nin tasarımındaki başlıca neden kullanım kolaylığıdır. Otomatik tespit etme, tak çalıştır özelliği ile ayar gerektirmemesi, istenildiği anda takılıp sökülebilmesi ve harici güç kaynağı gerektirmemesi gibi özellikleriyle kullanıcıya kullanım kolaylığı sağlamaktadır.

• **Güvenilirlik:** USB'nin güvenilirliği donanım tasarımından ve transfer protokollerinden kaynaklanır. Programlamaya ya da manüel bir müdahaleye gerek kalmadan hatalar tespit edilerek, vericiyi yeniden gönderim yapabileceği konusunda bilgilendirmektedir.

• **Maliyet:** Önceki arabirimlerden daha karmaşık bir yapıya sahip olmasına rağmen, USB'nin devre elemanları pahalı sayılmazlar. Özellikle düşük hız gerektiren cihazlar için son derece düşük maliyetlerle çalışmak mümkündür.

• **Esneklik:** USB, dört transfer tipi ve üç farklı hızı ile birçok cihaz için uygulanabilirlik kazanmaktadır. USB sayesinde transferler gerçek zamanlara yakın düzeyde gerçekleşir. USB protokolünde diğer iletişim standartlarından farklı olarak sinyallere fonksiyonlar yüklenmez.

• **Güç Tüketimi:** USB harici bir güç kaynağı gerektirmez. Tasarruflu devreler ve kodlar yardımıyla kullanılmayan cihazların gücü kesilir ve uyku moduna alınır.

• **Hız:** USB, geleneksel seri ve paralel iletişime oranla çok daha yüksek bir veri iletişimi sağlamaktadır. USB 2.0 üç hızı destekler: Yüksek hız (480Mb/s), Tam hız (12Mb/s) ve Düşük Hız (1,5Mb/s). USB 3.0 ise 5Gbps'ye varan hızlarda veri iletişimi sağlar. Çizelge 2.2'de çeşitli arabirimlerle USB'nin hız karşılaştırılması gösterilmiştir:

**Çizelge 2.2.** USB ile diğer arabirimlerin karşılaştırılması (Hewlett Packard 2011)

| Arabirim | Format | Cihaz Sayısı (mak.) | Hız (mak.) bit/saniye | Kullanım |
|---|---|---|---|---|
| USB | Asenkron seri | 127 | 1.5M, 12M, 480M, 5G | Fare, Klavye, Modem, v.b. |
| RS-232 | Asenkron seri | 2 | 20K (115K bazı cihazlarda) | Modem, fare, enstrumantasyon |
| RS-485 | Asenkron seri | 32 | 10M | Veri toplama ve kontrol sistemi |
| IrDA | Asenkron seri | 2 | 115K | Yazıcı, dizüstü |
| Microwire | Asenkron seri | 8 | 2M | Mikrodenetleyici haberleşmesi |
| SPI | Asenkron seri | 8 | 2.1M | Mikrodenetleyici haberleşmesi |
| $I^2C$ | Asenkron seri | 40 | 3.4M | Mikrodenetleyici haberleşmesi |
| IEEE-488 | Paralel | 15 | 8M | Enstrumantasyon |
| Ethernet | Seri | 1024 | 10M/100M/1G | PC Network |
| MIDI | Seri Akım | 2 | 31.5K | Müzik, Görüntü kontrolü |
| Paralel Printer Port | Paralel | 2 | 8M | Yazıcı, tarayıcı, disk sürücüleri |

USB'nin tasarımcıya sağladığı avantajlar ise şunlardır:

• **İşletim Sistemi Desteği:** USB'ye destek veren bir sitemin üç asgarisi vardır: Sisteme takılan/sökülen cihazı algılamak, takılan cihazlarla haberleşmek ve son olarak yazılım sürücülerinin ve uygulamaların USB ile senkronize çalışmasını sağlamaktır.

• **Çevre birim Desteği:** USB'li cihazların donanımında bir kontrolör bulunur, istekleri yanıtlamak ise çevre birimine düşer.

• **USB–IF Forum:** USB–IF kâr amaçlı olmayan, bilgi, yazılım ve donanım aletleri ile test olanakları sağlayan bir kuruluştur.

USB'nin gerek kullanıcıya gerekse tasarımcıya sağladığı avantajların yanı sıra dezavantajlarından da kısaca bahsetmek gerekir. Bu dezavantajlar kullanıcı ve tasarımcı açısından maddeler halinde aşağıdaki gibi sıralanabilir:

• **Kullanıcıya Yönelik sorunlar:** Eski donanımlar ve işletim sistemleriyle uyumlu olmaması en temel dezavantajdır. Bundan başka, hız ve mesafe sınırlamaları nedeniyle USB kullanımı bazı durumlarda pratik olmamaktadır.

• **Eski Donanımlarla Uyumsuzluk:** Eski donanımlarda USB portu yoktur. USB olmayan bir cihazı USB porta bağlamanın yollarından biri konvertör kullanmaktır. Ancak bu çözüm sadece konvertör sürücüsünün tanıdığı konvansiyonel protokollerin kullanıldığı çevre birimlerinde olumlu sonuç verir.

• **Sürat Limitleri:** USB çok yönlü bir arabirim olmasına rağmen, USB ile her şeyi halletmeye çalışmak doğru değildir. Çizelge 2.1'e göre rakibi FireWire'dır. Yüksek hızlı USB, IEEE–1394 ile yarışabilirken IEEE–1394b ile yarışabilecek bir seçenek sunamamaktadır. Ancak FireWire'a göre yavaş olmasına rağmen daha ucuz olması USB'nin bir avantajı olarak görülebilir.

• **Mesafe Sınırlamaları:** USB masaüstü yol olarak tasarlanmıştır. USB için kablo boyu 5 metreye kadar çıkabilir. Oysa RS–232, RS–485 veya Ethernet daha uzun mesafelere izin verir. Bununla birlikte beş adet göbek (hub) kullanarak mesafeyi 30 metreye çıkarmak mümkündür.

• **Cihazlar Arası Haberleşme:** Masaüstü tasarımın bir sonucu haberleşmenin karargâh tarafından denetlenmesidir. Yani cihazlar doğrudan haberleşemezler.

• **Tasarımcıya yönelik sorunlar:** Programlamanın karmaşıklığı başlıca dezavantajdır. Gerek PC'nin ve gerekse cihazın donanımında gözden kaçan noktalar (bug'lar) ciddi sorunlar yaratır.

• **Protokolün Karmaşıklığı:** USB çevre biriminin programlanması, protokoller hakkında ayrıntılı bilgi sahibi olmayı gerektirir. Haberleşmenin büyük kısmı yonga tarafından otomatik olarak halledilir. Ancak sürücü yazanların, protokolleri ve sürücünün işlevini iyice anlamış olmaları gerekmektedir.

• **İşletim Sistemi Desteğinin Artırılması:** Windows sınıf sürücüleri, uygulamaların cihazlarla haberleşmesini sağlar. Cihaz bu sürücülerle çalışmadığı takdirde bir sürücü yazma problemi ortaya çıkabilir.

• **Donanım Sorunları (Bug'lar):** İlk karargâh denetleyicilerinin donanımları sorunlu olabiliyordu ve aynı şekilde çevre birimlerinde de sorunlarla karşılaşılabiliyordu. Bundan kaçınmanın en iyi yolu donanım seçiminde gereken dikkati göstermek ve son bilgileri elde etmektir.

• **Bedeller:** USB–IF tarafından sunulan belirtim (spesifikasyon), web sitesinden ücretsiz olarak alınabilir ve USB yazılımı geliştirmek için lisans bedeli ödemek gerekmez. Ancak USB arabirimi satışlarında durum farklıdır. Belirli bir Üretici Kimliğine sahip olmayı gerektirir ve bunun da belli bir bedeli vardır (Axelson 2005).

### 2.1.2. USB protokolü

USB seri, paralel vb. gibi diğer arabirimlere göre daha karmaşık bir iletişim protokolüne sahiptir. Bu yüzden USB arabirimi ile dizayn edilecek bir cihazın belirli gereksinimlerinden söz edilecektir.

#### 2.1.2.a. USB bileşenleri

USB'nin başlıca bileşenleri şunlardır:

• USB Karargâhı (Host): USB karargâh platformu, USB karargâh denetleyicisinin kurulduğu yer ile alıcı yazılım/cihazın çalıştığı yerdedir. USB Karargâh Denetleyicisi, karargâh ile USB çevre birimleri arasında bir arabirimdir. Karargâh; USB cihazların takılıp sökülmesi, karargâh ile cihazlar arasındaki veri akışının ve kontrolünün yönetimi, takılan cihazlara güç sağlanması ve daha birçok işlemi yürütmekle görevlidir.

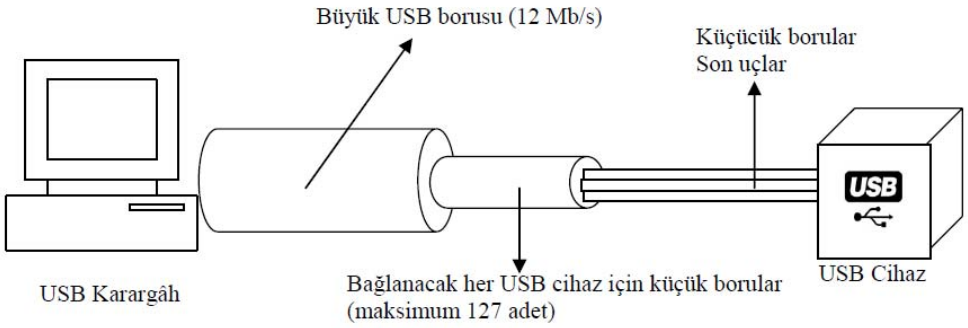
• USB Göbek (Hub): Birçok USB cihazın tek bir USB portuna takılarak, USB karargâhıyla haberleşebilmesi için kullanılan cihazlara göbek (hub) denir. Karargâhın arka alanında bulunan göbekler kök göbek (root hub) olarak adlandırılır. Diğer göbekler ise harici göbekler (external hubs) olarak adlandırılır.

• USB Cihaz (Device): USB yolu üzerinden veri ya da kontrol bilgisi ileten veya alan cihazlara USB cihaz adı verilir (Şahin vd 2006).

#### 2.1.2.b. USB cihazlarda veri akışı

USB, karargâhla cihazların birbirleri arasında haberleşebilmeleri için bir protokol sağlar. Karargâh belirli bir zamanda sadece bir USB cihazı ile haberleşebilir, yani aynı anda birden çok cihazla haberleşme durumu söz konusu değildir. USB cihazların karargâhla olan haberleşmeleri, karargâh üzerindeki yazılım hafıza tamponları ile

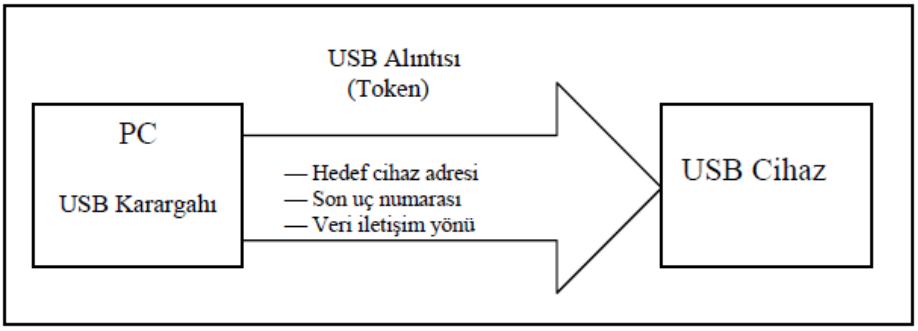
cihazların son uçları arasına kurulan sanal borular yordamıyla birebir gerçekleştirilir. USB yolundaki veri akışı yarı çift-yönlüdür (half-duplex), yani belirli bir zaman zarfında veri sadece bir yönde taşınabilir (Tan 1997).




**Şekil 2.1.** USB boruları ve son uç kavramı (Tan 1997)

Şekil 2.1'de, USB karargâhının kendisine bağlanacak USB çevre birimlerinin her birine (127 adete kadar) küçük borular gönderdiği şematiksel olarak gösterilmiştir. USB alıntısı (token) içerisinde 7 adet adres bit'i bulunmaktadır. Ancak, USB varsayılan (default) adresi olarak adlandırılan, 0000 000B adresi yeni takılan her cihaz için onu yapılandırmak ve kontrol etmek için kullanılır.

Şekil 2.1'de gösterilen her küçük boru, 16 çift (16 adet IN ve 16 adet OUT) toplamda 32 küçücük boruya (son uç) kadar ayrılabilir. Küçük borular, duraksız (stream) ve mesaj (message) boruları olmak üzere iki çeşittir. Duraksız borular eşzamanlı, kesme ve yığın tipi veri iletişiminde kullanılırken, mesaj boruları sadece kontrol veri iletişiminde kullanılır. Her USB alıntısı 4 adet son uç bitine sahiptir ve bu alıntı IN (giriş) veya OUT (çıkış) alıntısı olarak tanımlanır. Eğer bir USB cihazı; karargâhtan bir IN alıntısı alırsa, karargâha veri gönderecektir, karargâhtan OUT alıntısı alırsa, karargâh veri gelmesini bekleyecektir (Tan 1997).




**Şekil 2.2.** USB veri hareketi

Son uç, karargâhla cihaz arasında yapılan haberleşmede veri akışının kaynak tarafında ya da sonlanma tarafında bulunan bir tampon olarak düşünülebilir. Her USB cihazı birbirinden bağımsız son uç koleksiyonlarından oluşur. USB'nin tüm hız opsiyonları için (düşük, tam ve yüksek) bir çift-yönlü kontrol son uç (Son Uç 0) ve 15 tek yönlü son ucu destekler. Her tek-yönlü son uç gelen veya giden transfer işleminde kullanılabilir.

### 2.1.2.c. USB veri transfer tipleri

USB cihaz, karargâh üzerindeki hafıza tamponu ile cihaz üzerindeki son uç arasındaki boru üzerinden karargâh ile haberleşir. USB spesifikasyonu dört veri transfer tipini desteklemektedir. Belirli bir son uç için veri transfer tipi seçimi cihazın ve yazılımın gereksinimlerine göre seçilmektedir. Bu seçim son uç tanımlayıcıda seçilmektedir (Axelson 2005).

**- Eşzamanlı (Isochronous) transfer**

Büyük miktarlardaki verinin (1023 bayt'a kadar), bilgisayar ile çevre birim arasında periyodik ve sürekli bir şekilde iletilmesini sağlayan transfer tipidir. Zaman açısından kritik olan ama hataya karşı hassas olmayan akan veri (streaming data) iletiminde kullanılır. Örneğin USB kamera ya da ses kartları gibi. Sabit bir bant genişliği sağlanarak küçük hatalara izin verilirken veri akışının kesilmemesi sağlanır.

Eşzamanlı borusu tek yönlü bir borudur, yani belirli bir son uç sadece veri iletir veya veri alır. İki yönlü bir eşzamanlı iletişim isteniyorsa, her iki yön için birer adet son uç kullanılmalıdır (Axelson 2005).

**- Yığın (Bulk) tipi transfer**

Yığın tipi transfer ise büyük miktarlardaki verinin aktarımı için kullanılan bir veri transfer tipidir. Taşınacak verinin alıcı kısmında doğru bir biçimde alınması garanti edilirken, verinin iletim zamanı konusunda garanti verilemez. Tipik olarak, yazıcı ve tarayıcı haberleşmesinde yığın tipi iletişim kullanılır.

Diğer akış borularında olduğu gibi, yığın boruları da tek yönlüdür, yani çift yönlü bir yığın iletişimi için iki adet yığın son ucuna ihtiyaç vardır.

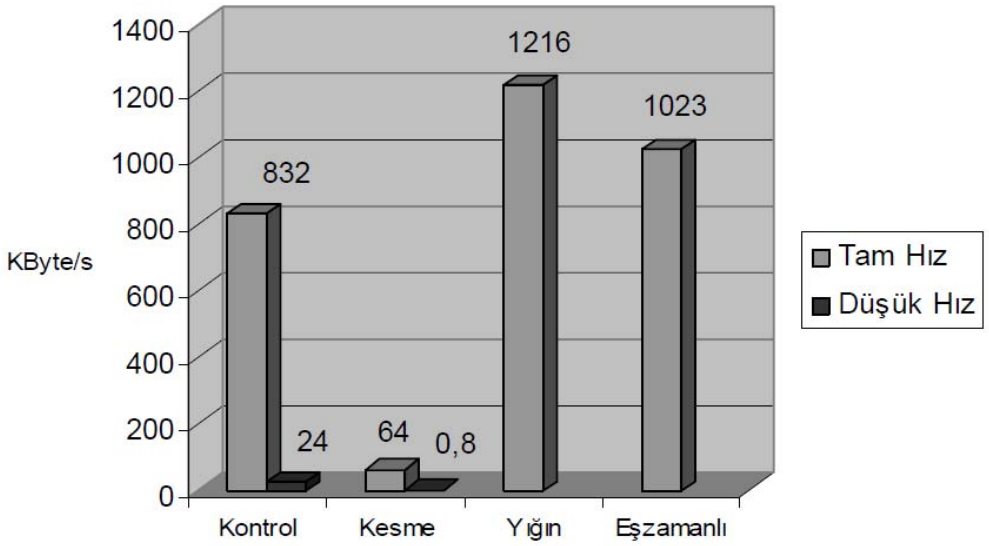
Yığın tipi iletişim diğer iletişim tiplerinden farklı olarak sadece tam-hız ve yüksek-hız USB cihazlarda kullanılabilir. Yığın son uçları için maksimum veri paketi boyutu, tam hızlı cihazlar için 8, 16, 32 veya 64 bayt, yüksek hızlı cihazlar için ise sadece 512 bayttır (Axelson 2005).

**- Kesme (Interrupt) tipi transfer**

Ani reaksiyon gerektiren fare ya da klavye ile iletişimde kullanılır. Diğer veri transferlerine göre daha düşük miktarlardaki verinin, zaman ve verinin doğruluğu garanti altına alınarak iletilmesi için kullanılır. Örneğin, bilgisayara bağlı bir fare ile yapılan veri iletişiminde kullanılabilir. Kesme borusu da eşzamanlı boru gibi tek yönlü ve periyodik bir borudur. Kesme son uçları için maksimum veri paketi boyutu, düşük hızlı cihazlar için 8 bayt veya daha az, tam hızlı cihazlar için 64 bayt veya daha az ve yüksek hızlı cihazlar için ise sadece 1,024 bayt veya daha azdır (Axelson 2005).

**- Kontrol (Control) tipi transfer**

PC ile USB arasında kurulum, ilk değerlerin atanması ve konfigürasyon bilgilerinin taşınmasında kullanılır. Bu bilgiler önemli olduğundan CRC denilen kontrol kodları eklenir. Kontrol borusu çift-yönlü bir boru olup, her iki yönde veri akışını desteklemektedir. Kontrol son uçları için maksimum veri paketi boyutu, düşük hızlı cihazlar için 8 bayt, tam hızlı cihazlar için 16, 32 veya 64 bayt ve yüksek hızlı cihazlar için ise sadece 64 bayttır.



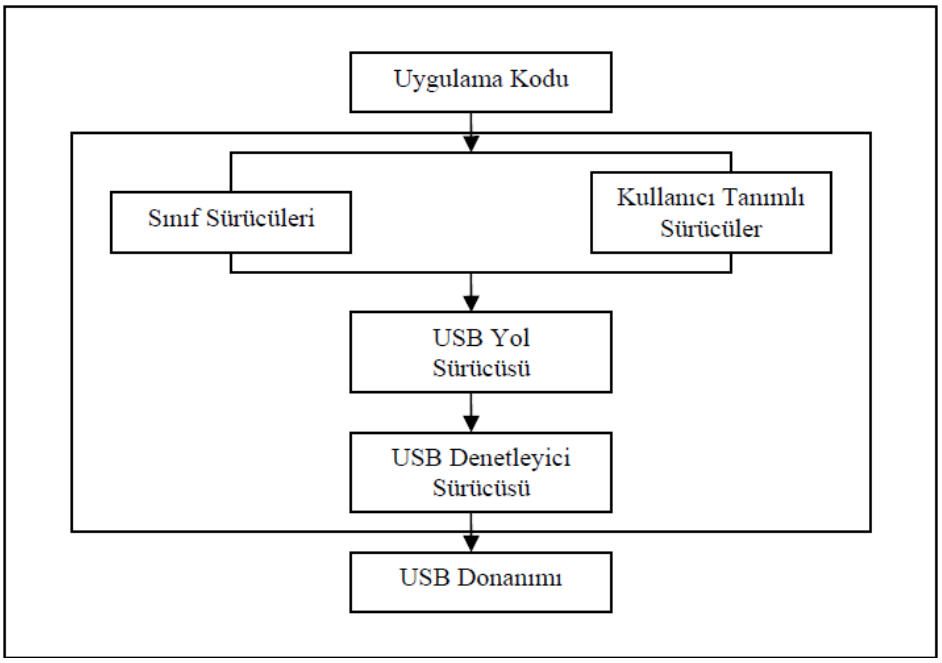

**Şekil 2.3.** USB veri iletişim tiplerinin teorik olarak taşıyabilecekleri maksimum veri büyüklüğü

USB veri iletişim tiplerinin taşıyabilecekleri veri miktarı açısından karşılaştırılması Şekil 2.3'te grafiksel olarak verilmiştir (Axelson 2005).

**2.1.2.d. USB sürücüleri**

Win32 sürücü modeli (WDM, Win32 Driver Model) Windows sürücü sisteminin temelini oluşturur. Bu modelde donanım ile iletişim işlemi çeşitli katmanlara ayrılmıştır. Uygulama katmanında yazılım ile ilgilenenlerin tanıdığı API (Application

Program Interface) çağrıları yer alır. API çağrıları WDM içerisinde yer alan hazır tanımlanmış sınıfları ya da kullanıcının tanımlamış olduğu sürücü fonksiyonlarını çağırır. Buradan da daha alt seviye giriş-çıkış istek paketleri (I/O request packets, IRP) kullanarak gerçekleştirilir. Bir alt kademede bulunan USB yol sürücüsü ise USB cihazlarına ilişkin güç, sayma (enumeration) ve çeşitli bilgi alışverişi işlemlerini meydana getirir. Daha aşağıdaki USB denetleyici sürücüsü ise bilgisayar içerisindeki USB donanımına doğrudan erişimi sağlar. Bu sürücüler Windows içerisinde mevcuttur. Şekil 2.4'te USB sürücü modeli gösterilmektedir.




**Şekil 2.4.** USB sürücü modeli

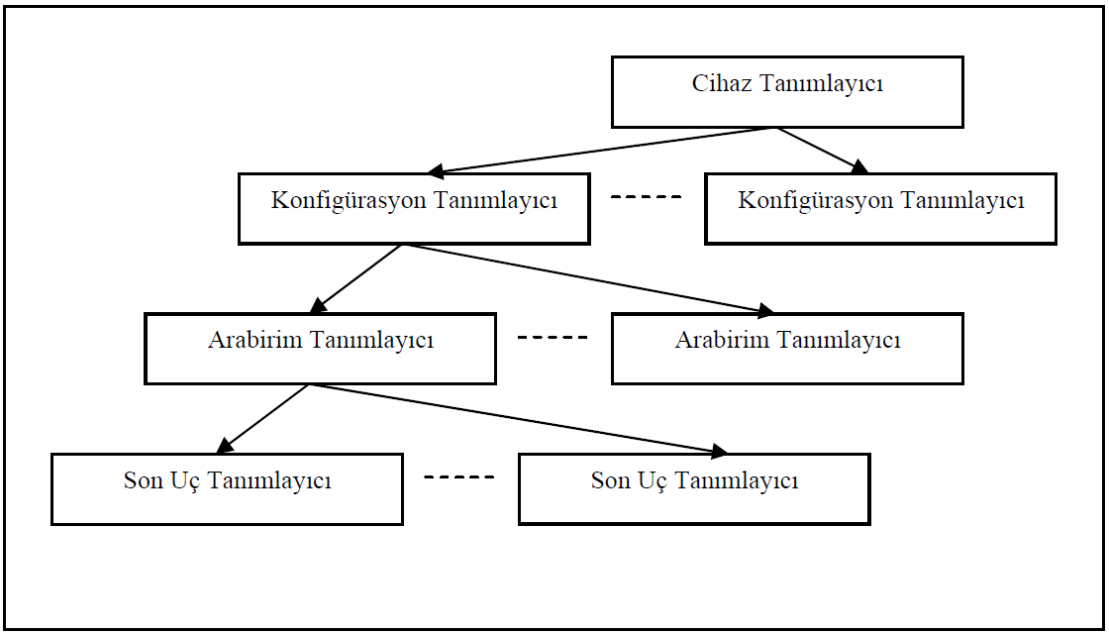
### 2.1.2.e. USB konfigürasyonu

Bir USB fonksiyonu başlatılmadan önce, cihaz konfigürasyonu yapılmalıdır. Karargâh, yapılandırmayı, USB cihaz üzerinden veri sağlama yoluyla almış olduğu bilgilerle gerçekleştirir. USB cihazlar, işlevlerini tanımlayıcılar (decriptors) ile karargâha rapor ederler. Tanımlayıcı, hangi verinin hangi formatta transfer edileceğini belirtir.

USB tanımlayıcılar hiyerarşik bir yapı içerisinde dört kısımda incelenebilir:

• Cihaz tanımlayıcıları
• Konfigürasyon tanımlayıcıları
• Arabirim tanımlayıcıları
• Son uç tanımlayıcıları

Her USB cihaz için sadece bir adet cihaz tanımlayıcısı vardır. Şekil 2.5'te gösterildiği gibi her cihaz bir veya daha çok konfigürasyona, her konfigürasyon bir veya daha çok arabirime ve her arabirim sıfır veya daha çok son uca sahiptir (Microchip Technology Inc. 2004).




**Şekil 2.5.** Tanımlayıcı hiyerarşisi

**- Cihaz tanımlayıcısı**

Cihaz tanımlayıcısı, üretici kimliği (VID, Vendor Identity), ürün kimliği (PID, Product Identity), seri numarası, cihaz sınıfı ve konfigürasyon sayısı gibi genel bilgiler verir. Daha önce de belirtildiği gibi her cihaz için sadece bir cihaz tanımlayıcı vardır.

**- Konfigürasyon tanımlayıcısı**

Konfigürasyon tanımlayıcı, cihazın güç gereksinimleri ve belirli bir konfigürasyonda kaç farklı arabirimin destekleneceği bilgilerini verir. Bir cihaz için birden fazla konfigürasyon olabilir, örneğin düşük-güç ve yüksek-güç konfigürasyonu gibi.

**- Arabirim tanımlayıcısı**

Arabirim tanımlayıcısı, arabirimin sınıfı yanında arabirimde kullanılacak son uç sayısının detaylarını da verir. Bir konfigürasyon için birden fazla arabirim olabilir.

**- Son uç tanımlayıcısı**

Son uç tanımlayıcı, son ucun transfer tipi ve yönü ile diğer spesifik özelliklerini verir. Bir USB cihazı birçok son uca sahip olabileceği gibi, son uçlar farklı konfigürasyonlar tarafından paylaşılmış olabilir.

## 2.2. GPIB

### 2.2.1 Tanım

Günümüzde GPIB (General Purpose Interface Bus – Genel Amaçlı Arayüz Yolu) olarak bilinen iletişim standardı, orjinali 1965'te Hewlett Packard tarafından test cihazlarına

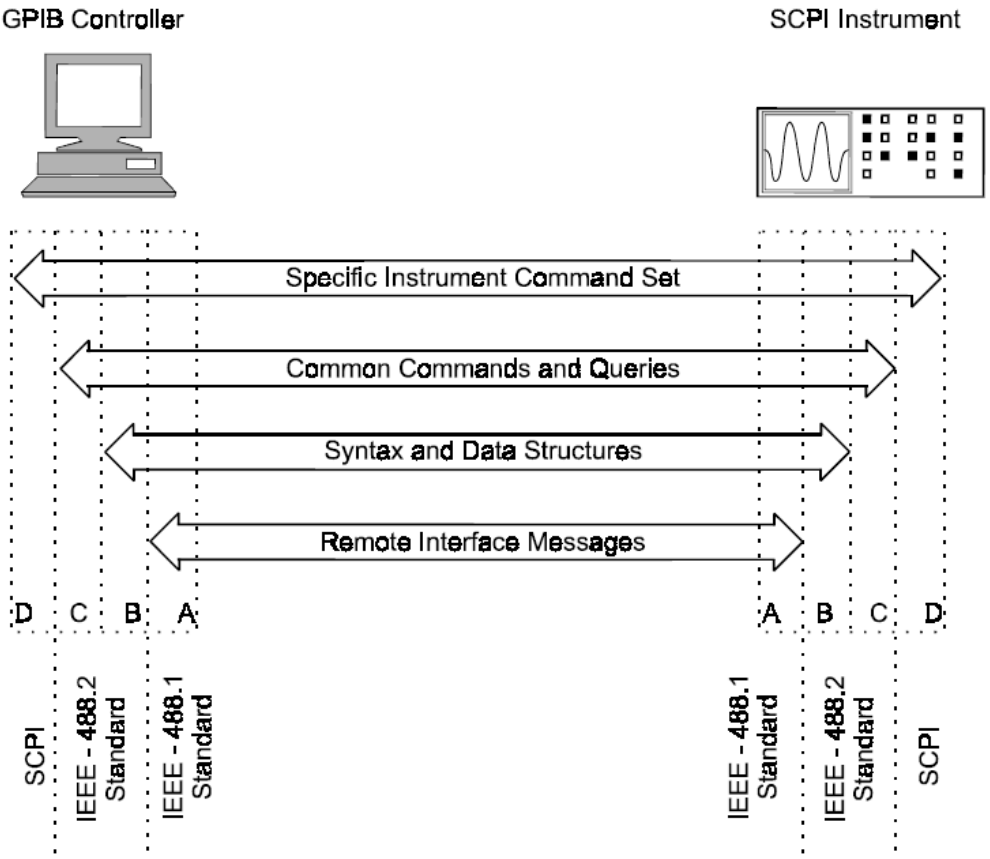
bağlanmak ve onları kontrol etmek için geliştirilmiştir. GPIB, programlanabilir cihazları bilgisayarlara bağlamadaki kullanışlılığı, esnekliği ve hızından dolayı geniş çapta kabul gördü ve farklı üreticiler tarafından kendi programlanabilir cihazları için uygulandı.

Farklı üreticilerin dijital kontrol cihazları ve programlanabilir cihazları tanıtması ile birlikte, cihazların birbirleriyle iletişimi için yüksek hızda iletişim arayüzü standardı gerekliliği çok çabuk anlaşıldı. IEEE komitesi tarafından yapılan bir çalışma sonucu IEEE 488-1975 standardı 1975 yılında yayınlandı. Küçük değişikliklerle 1978 yılında Avrupa'da IEC-625 olarak standartlaştırıldı. Revize standart Avrupa'da daha yaygın olarak kullanılmıştır. IEC-625, 1987 yılında IEEE 488.1-1987 standardı olarak standartlaştırıldı. Bu standart iletişim arayüzünün elektrik, mekanik ve donanım protokol şartnamelerini tanımlayarak programlanabilir cihazların bağlantısını çok basitleştirdi. İlk başlarda farklı üreticilerin cihazları standart bir kablo ile bağlanırdı.

1987 yılında veri formatları, durum raporlama, kontrolör işlevselliği, hata işleme ve ortak komutları tanımlayan IEEE-488.2 standardı yürürlüğe girdi. IEEE-488.2 standardı yazılım protokolleri konusuna ve donanım odaklı IEEE-488.1 standardına uyan cihazlarla olan uyumluluğu korumaya yoğunlaşmıştır.

1990'da bir grup cihaz üreticisi IEEE-488.2 standardının genişletilmiş başka bir hali olan, IEEE-488.2 standardını temel alan ve herhangi bir donanım bağlantısı ile kullanılabilen programlanabilir cihazlar için kullanılan ortak komut seti tanımlayan, SCPI standardını duyurdu (Park and Mackay 2003).

Standartlar arasındaki ilişki Şekil 2.6'da gösterilmiştir.



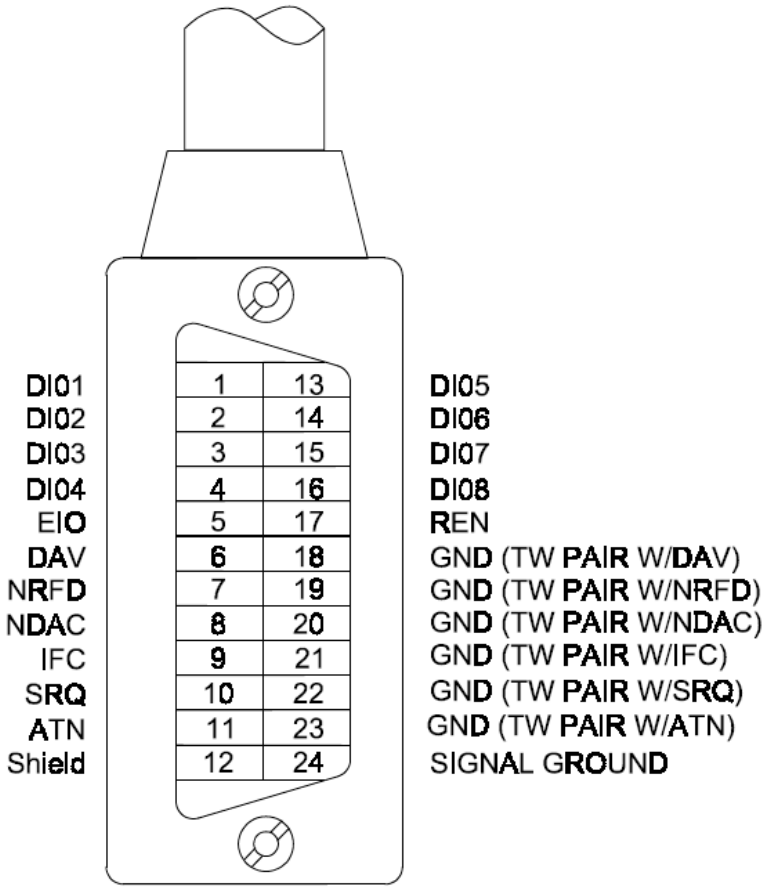

**Şekil 2.6.** IEEE-488.1, IEEE-488.2 ve SCPI Standardı (Park and Mackay 2003)

GPIB, 15 cihazın veya bir ortak paralel veri iletişim yolundaki cihazların eşzamanlı olarak bağlanmasına izin veren bir arayüz tasarımıdır. GPIB, cihazların kontrol edilmesine veya verilerin kontrolöre, yazıcıya veya çiziciye iletilmesine olanak sağlar. GPIB, verinin düzenli iletilmesinin, farklı cihazların adreslenmesinin ve standart yol yönetim komutlarının yöntemlerini ve arayüzün fiziksel detaylarını tanımlar. Takip eden bölümlerde bu konular anlatılmıştır.

### 2.2.2. Elektriksel ve mekaniksel karakteristikler

GPIB her birinin sonunda standart konnektör bulunan korumalı 24 kablo ile verileri taşır. GPIB konnektörüne yeni bir cihaz eklemek veya çıkarmak yola yeni bir kablo eklemek ile mümkündür. Vidalar  güvenli bir şekilde her bir konnektörün birbirine

bağlanmasını sağlar. Şekil 2.7'de GPIB konnektörü ve pinlerine karşılık gelen hatlar verilmiştir (Park and Mackay 2003).




**Şekil 2.7.** GPIB Konnektörü ve Pin Bağlantıları (Park and Mackay 2003)

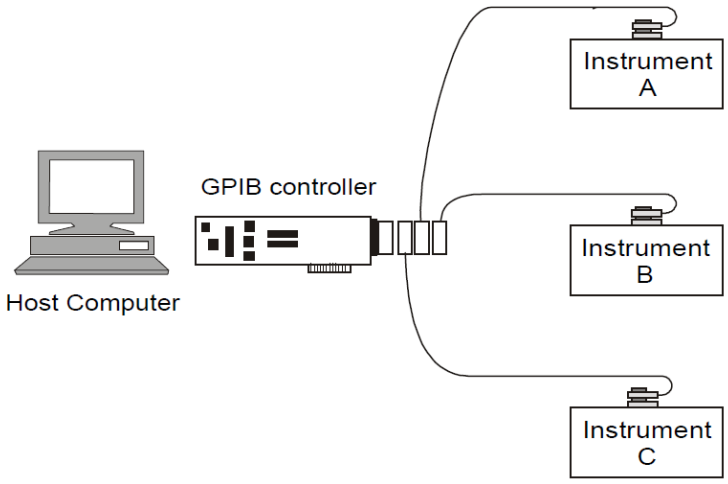
GPIB, 8 veri hattı ve 8 çift kontrol ve yol yönetim hattı olmak üzere 24 hattan oluşur. Veri hatları paralel bir yapılanmada yol boyunca tek seferde bir bayt olmak üzere sadece veri taşırlar. Kontrol ve yol yönetim hatları veri akışını senkronize eden çeşitli yol yönetim görevlerinde kullanılır. Yola veri ya da komut gönderildiğinde yol yönetim hatları bu ikisini ayırt eder. GPIB yolda taşınan bilgileri ikilik voltaj sinyalleri kullanarak gösterir. Bu voltajın hattaki iki durumunu göstermede "true" ve "false" sembollerini kullanır. GPIB, düşük voltaj durumunda "true" ve yüksek voltaj durumunu "false" olarak gösteren, negatif lojik kuralını kullanır. Standard TTL voltaj seviyeleri

kullanılır. Örneğin, bir hat “true” ise TTL voltaj seviyesi düşüktür (≤0.8 V) ve hat “false” ise TTL voltaj seviyesi yüksektir (≥2.0 V). “Low-true” lojiğinin anlamı herhangi bir cihaz yol voltajını “low-true” yapabilir, ancak hiçbir cihaz o hattaki bütün cihazlar “high=false” olmadıkça yol voltajını “high=false” yapamaz.

### 2.2.3. Fiziksel bağlantı ayarları

GPIB üzerindeki cihazlar yıldız ve linear olarak bağlanabilir. Bunlar Şekil 2.8 ve Şekil 2.9’da gösterilmiştir.

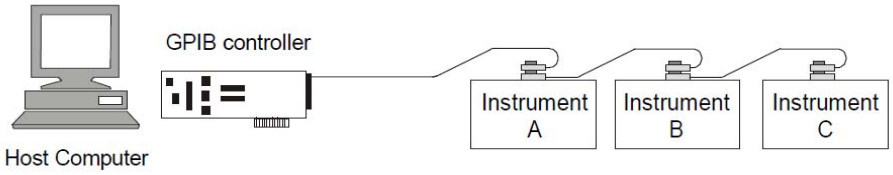
Bir yıldız yapılanmada cihazlar ayrı bir GPIB kablosu aracılığı ile direk olarak kontrolöre bağlanır. Bütün konnektörler kontrolördeki aynı porta bağlanırlar. Bu yapılanmanın bir dezavantajı kabloların boy sınırlaması nedeniyle yol üzerindeki bütün cihazlar kontrolöre yakın olmalıdır.




**Şekil 2.8.** GPIB Yıldız Yapılandırma (Park and Mackay 2003)

Zincir yapılanmada kontrolör de dahil bütün cihazlar yanındakine sıra ile bağlıdır. Kontrolör en başta veya en sonda olmak zorunda değildir ve herhangi bir yere bağlanabilir. Kontrolör bir anlamda yol üzerindeki olayları koordine eder. Bu yapılanma

donanım eklemek için en uygun yöntemdir. Bu yapılanmanın bir dezavantajı bir cihaz veya onun kablosu çıkarılırsa yazılımın tekrar yapılandırılması gerekebilir (Park and Mackay 2003).




**Şekil 2.9.** GPIB Zincir (doğrusal) Yapılandırma (Park and Mackay 2003)

Bağlantılar, yıldız ve lineer yapılanmalar önerilmesine rağmen aşağıdaki şartlar sağlanmak koşulu ile başka yöntemlerle de yapılabilir. Bağlantı şartları;

- Tüm cihazlar yola bağlı olmalıdır
- Kontrolör de dahil 15 cihazdan fazla cihaz bağlanmamalıdır ve cihazların en az 2/3'si açık olmalıdır.
- Herhangi iki cihaz arasındaki kablo uzunluğu 4 metreyi geçemez.
- Toplam kablo uzunluğu 20 metreyi geçmemelidir.

GPIB üzerindeki bir cihaz diğer 14 cihaza veri gönderebilir. Çünkü GPIB asenkron el sıkışma (handshaking) kullanır. Veri iletim oranı donanım arayüzünden çok cihazların kendi iletim oranlarına bağlıdır (Park and Mackay 2003).

### 2.2.4. Cihaz tipleri

Bir veri iletimi açısından bir GPIB arayüzü üzerinde iletişim kurabilecek 4 farklı cihaz grubu vardır.

### 2.2.4.a. Konuşmacı

Bir konuşmacı bir dinleyici cihaza sadece veri gönderebilen tek yönlü iletişim cihazıdır. Veri almaz. Konuşmacı kontrolörden bir sinyal bekler ve daha sonra veriyi yol üzerine koyar. Aynı anda sadece bir konuşmacı konuşabilir. Genel örnekler basit dijital voltmetreler, A/D dönüştürücüler.

### 2.2.4.b. Dinleyici

Bir dinleyici, diğer cihazdan sadece veri alabilen tek yönlü iletişim cihazıdır. Veri gönderemez. Kontrolör yolu okuması için sinyal gönderdiğinde veriyi yoldan alır. Genel örnekler yazıcılar, çiziciler, kaydedicilerdir.

### 2.2.4.c. Konuşmacı/Dinleyici

Bir konuşmacı/dinleyici konuşmacı ve dinleyicinin her ikisinin karakteristiğinin karışımına sahiptir. Ancak, asla aynı anda bir konuşmacı ve bir dinleyici olamaz. Yaygın bir örnek programlanabilir dijital voltmetre. Dijital voltmetre (DVM) onun aralığı ayarlandığında dinleyici olabilir ve bir konuşmacı olarak ayarlandığı zaman sonuçları kontrolöre gönderebilir.

### 2.2.4.d. Kontrolör

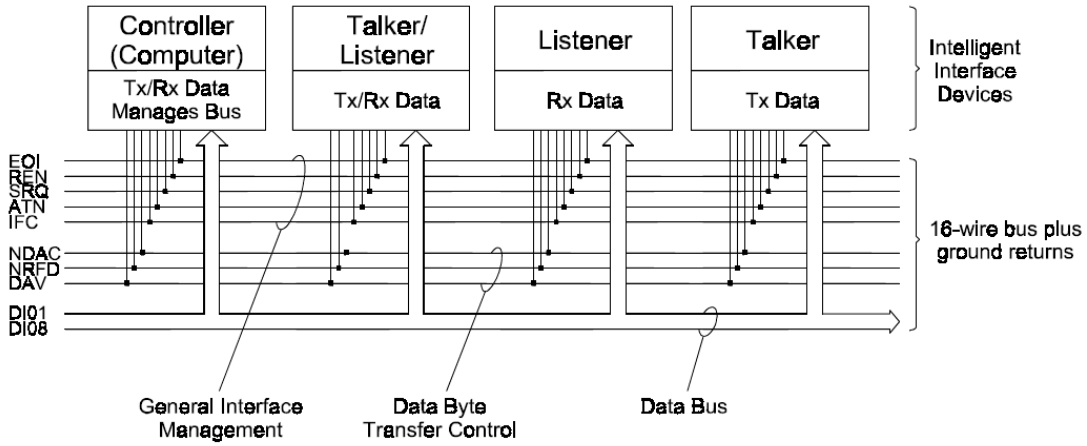
Bir kontrolör GPIB üzerinde olan herşeyi yönetir ve kontrol eder. O genelde PC veya mikroişlemci kontrollü cihaz gibi yetenekli ya da programlanabilir bir cihazdır. Hangi cihazın ne zaman veri göndereceğini (konuşmacı) ve hangi cihazın ne zaman veri alacağını (dinleyici) karar verir. Herhangi bir GPIB uygulamasında karışıklıktan kaçınmak için sadece bir aktif kontrolör olabilir. Bu kontrolör aktif/görevde olan kontrolör olarak adlandırılır (controller in charge-CIC). Birkaç tane kontrolör olabilir, ancak karışıklığı önlemek amacıyla herhangi bir anda sadece biri aktif olur (CIC).

Kontrolör aynı zamanda konuşmacı ve dinleyicinin özelliklerine sahiptir. Bazı durumlarda birden çok PC'nin GPIB üzerinde eşzamanlı bağlantısında, PC'lerden biri kontrolör olarak ayarlanır diğerleri konuşmacı ve dinleyici olarak ayarlanır.Kontrolör verinin her iletiminden haberdar olmalıdır. Konuşmacının dinleyiciye bir mesaj gönderebilmesi için öncelikle konuşmacının ve dinleyinin bir adresinin olması gerekir. Mesaj gönderildikten sonra kontrolör her iki cihazın adresini kaldırır. Bazı GPIB yapılanmaları kontrolöre ihtiyaç duymaz. Örneğin bir konuşmacı bir veya birden fazla dinleyiciye bağlanmışsa.

Aktif veya adreslenmiş olan konuşmacı veya dinleyinin değiştirilmesi gerektiği durumda bir kontrolöre ihtiyaç duyulur. Bu durumda bir cihaz yolda onun konuşma adresini görür, onun bir konuşmacı olarak davranıp veri göndermesi gerektiğini bilir. Tersi durumda bir cihaz yolda onun dinleme adresini gördüğünde bilir ki o dinleme eğiliminde olup bu nedenle veri bekler (Park and Mackay 2003).

### 2.2.5. Yol yapısı

Şekil 2.10'da gösterilen GPIB arayüz sistemi 16 sinyal hattı ve 8 toprak hattından oluşmaktadır.




**Şekil 2.10.** GPIB Yol Yapısı (Park and Mackay 2003)

16 sinyal hattı, 8 veri hattından ve 8 kontrol hattından oluşur. 8 kontrol hattının 3 tanesi veri iletimini koordine eden (DAV, NRFD ve NDAC) el sıkışma (handshaking) hattıdır. Kalan 5 hat ise yol kontrol ve yönetimi (ATN, REN, IFC, SRQ ve EOI) için kullanılır. 8 toprak hattı ise elektronik koruma sağlar ve yol kontrol sinyallerinin birbirlerinden ve harici sinyallerden etkilenmesini engeller.

8 veri hattı (DI01-DI08) hem veri mesajlarını hem de komut mesajlarını taşır. Bütün komutlar ve bir çok veri 8-bit veri aktarımında 7-bit ASCII kod kullanır. DI08, parity için kullanılır ya da kullanılmaz. Attention line (ATN-dikkat) hattının durumu bilginin komut mu yoksa veri mi olduğuna karar verir. Komut mesajları ATN hattı aktif edilerek, veri mesajları ise ATN hattı pasif edilerek gönderilir.

5 sinyal hattı GPIB'de bilgi akışını yönetir. Bunlar aşağıda açıklanmıştır.

- ATN (Dikkat)- Kontrolör veri hattından komut göndereceği zaman ATN hattını "true" yapar. Bütün cihazlar dinleyici olur ve iletişimin bir parçası olur. ATN hattı "false" yapıldığında yol üzerindeki bilgi veri mesajı olarak algılanır.

- IFC (Arayüz temiz)- Bu hat sadece yolu başlangıç ayarlarına getirmek ve aktif kontrolör (CIC) olmak için sistem kontrolörü tarafından kullanılabilir. IFC hattı GPIB'in ana reseti olup, aktif hale geldiğinde tüm cihazlar pasif hale geçer.

- REN (Uzaktan kontrol aktif) – Sistem kontrolörü, cihazları uzaktan kontrol konumuna getirmek için REN hattını kullanır. REN hattı aktif yapıldığında ve bir cihaz dinleyici olarak adreslendiğinde, bu cihaz uzaktan programlanabilir duruma geçer.

- SRQ (Servis isteği)- Herhangi bir cihaz servis isteğini CIC'e eş zamansız olarak göndermek için kullanır. CIC'in görevi SRQ hattını görütülemek, cihazların SRQ isteğini yoklamak ve ne tür servis ihtiyacı olduğuna karar vermektir.

- EOI (Bitir ya da tanımla) – EOI hattının iki amacı vardır. İlki bir konuşmacı mesajdaki verinin son baytını göstermek için EOI hattını aktif yapar. Bir dinleyici EOI hattı "true" olduğunda veri okumayı durdudur. EOI hattının ikinci amacı ise cihazların cevaplarını paralel bir şekilde vermelerini ister.

Üç el sıkışma (handshake) hattı cihazlar arasında mesaj baytlarının iletimini asenkronize şekilde kontrol eder. GPIB bu üç hattı veri hatlarındaki mesaj baytlarının hatasız olarak gönderilip ve alınmasını garanti etmek için kullanır.

- NRFD (Veri için hazır değil)- NRFD handshake hattı bir cihazın mesaj baytı almaya hazır olup olmadığını gösterir. Komutlar alındığı zaman bu hat konuşmacılar ve dinleyiciler tarafından kullanılırken, mesaj verileri alındığı zaman sadece dinleyiciler tarafından kullanılır.

- NDAC (Veri kabul edilmedi)- Bu hat cihazın mesaj baytını kabul edip etmediğini gösterir. NDAC komut alındığında tüm dinleyiciler ve konuşmacılar tarafından kullanılırken, mesaj verileri alındığında sadece dinleyiciler tarafından kullanılır.

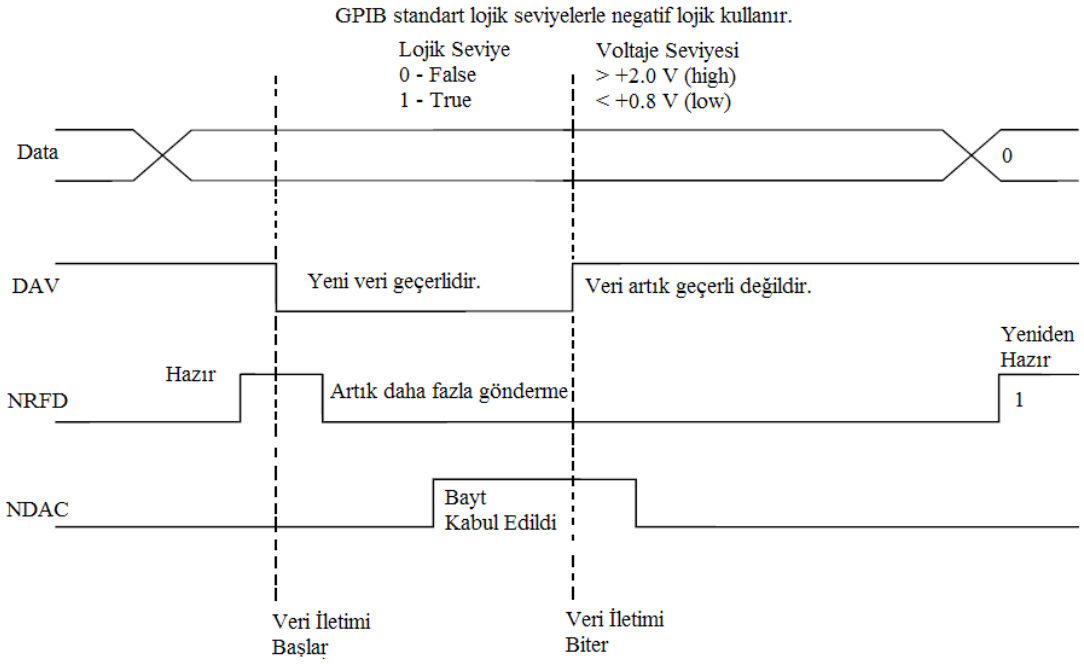
- DAV (Veri geçerli) – DAV handshake hattı veri hattı üzerindeki sinyallerin dengeli olduğunu ve bu yüzden geçerli olduklarını ve cihazlar tarafından kabul edilebileceklerini gösterir (Park and Mackay 2003).

### 2.2.6. GPIB el sıkışma (handshaking)

Aynı anda bir bayt olmak üzere veri GPIB paralel yüzünde eşzamansız iletilir. Verinin iletiminin koordinesi üç yol kontrol "handsake" hattındaki (DAV, NDAC ve NRFD) voltaj sinyalleri tarafından yapılır. Bu işlem three-wire interlocked handshake olarak adlandırılır. El sıkışma (Handshaking) bir konuşmacının, her bir dinleyici hazır olduğunda ve veriyi tüm dinleyiciler okuyana kadar diğer tüm dinleyiciler yolda tutarak, yola veri koyacağını garanti eder. Ayrıca el sıkışma (handshaking),

dinleyicilerin yolda sadece geçerli bir bayt hazır olduğunda veriyi kabul edeceğini garanti eder.

Konuşmacı ilk başta DAV'i pasif yapar. Daha sonra NDAC ve NRFD hatlarını izler. Konuşmacı yola herhangi bir veri koyabilmesi için NRFD hattının "high=false" olmasını beklemelidir. NRFD hattı dinleyiciler tarafından kontrol edilir. NRFD voltajı "high=false" olduğunda tüm dinleyiciler veri almak için hazırdır. Kısa bir gecikmeden sonra NRFD "high=false" olur, konuşmacı yol üzerindeki hazır olan geçerli veriyi göstermek için DAV'ı "low=true" yapar. Gecikme konuşmacının veri hattında kullandığı sürücü tipi tarafından belirlenir. Dinleyiciler DAV'da düşük voltajı saptadıkları anda veri hattındaki baytı okur ve artık daha fazla veri gönderilmemesi için NRFD'yi hemen aktif yapar. Her bir dinleyici veriyi kabul ettiğinde, NDAC pasif (high=false) olur. En son dinleyici de veriyi kabul ettikten sonra NDAC hattı "high=false" olur ve konuşmacıya verilerin kabul edildiğine dair sinyal gönderir. Sadece veri tüm dinleyiciler tarafından kabul edildiğinde, konuşmacı DAV voltajının "high=false" olmasına ve verisinin yoldan kaldırılmasına izin verir. Dinleyiciler bir sonraki döngü hazırlığı için veri iletiminin sona erdiğini göstererek NDAC'yi aktif yapar. El sıkışma (handshaking) zaman diyagramı Şekil 2.11'de gösterilmiştir (Park and Mackay 2003).




**Şekil 2.11.** GPIB Handshaking (El Sıkışma) Zaman Diyagramı (Park and Mackay 2003)

Unutulmamalıdır ki konuşmacı bir mesaj baytı göndermeden önce tüm dinleyicilerin daha önce gönderilen veriyi kabul etmesini bekler ve tüm dinleyicilerin hazır olduğundan emin olana kadar bir sonraki veriyi göndermez. GPIB'deki maksimum veri iletim hızı yoldaki en düşük hıza sahip dinleyici tarafından belirlenir.

### 2.2.7. Cihaz İletişimi

GPIB cihazları diğer bağlı bulunan cihazlarla paralel veri iletişim arayüzü içerisinde tek seferde bir bayt olmak üzere "cihaz bağımlı mesajları" ve "arayüz mesajlarını" birbirlerine göndererek iletişime geçer.

Cihaz bağımlı mesajları, cihazlarla ilgili programlama komutlarını, ölçüm sonuçlarını, cihaz durumu ve veri dosyalarını kapsayan özel bilgiler içerir. Bu mesajlar genelde veri mesajları olarak adlandırılır.

Arayüz mesajları, iletişim yolunun çalışmasını yönetir. Yolu başlangıca hazırlama, adresleme, cihazları pasif etme, uzak veya yerel programlama modlarını ayarlama gibi görevleri gerçekleştirir. Bu gibi mesajlar komut mesajları olarak adlandırılır. Komut mesajları ifadesi veri mesajlarını içeren cihaz bağımlı komutları ile karıştırılabilir. Örneğin, "*IDN?" cihazların tanımlanmasında kullanılan bir komut olmasına rağmen gönderilmesi itibari ile GPIB veri mesajları ile aynıdır (Park and Mackay 2003).

### 2.2.8. GPIB adresleme

GPIB'e bağlı her bir cihaz tek bir adrese sahiptir ve her bir cihazın tasarımı veri hattına indirilen veri veya komutların kendine mi yoksa diğer cihazlara mı gönderildiğini ayırt edecek hassasiyette olmalıdır. Cihaz adreslerinin atanması kullanıcı seçimine bağlı olup, genellikle cihazın arka tarafına yerleştirilmiş olan bir anahtar veya yazılımla belirlenir.

Kontrolör sistemdeki tüm cihazların adreslerini programında belirtir. Bir cihazın adresinin seçimindeki tek sınırlama, o adresin 0-30 arasında bir sayı olması gerekliliğidir.

GPIB'de iletişim için öncelikle kontrolörün uygun bir cihazı adreslemesi gerekir. Cihazı adreslemesi ve onun iletişim yolunda yer alması için kontrolör tarafından adres komut mesajı gönderilir. Çizelge 2.3'te adres komut mesaj formatı gösterilmiştir.

**Çizelge 2.3.** Adres komut mesaj formatı (Park and Mackay 2003)

| **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
|---|---|---|---|---|---|---|---|
| | TA | LA | X | X | X | X | X |

Bit 0 ve bit 4 cihazın GPIB birincil adresini ikilik (binary) sayı sisteminde temsil eder. Eğer bit 5 aktif edilmişse birincil adres olarak tanımlanmış cihaz dinleyici olarak, bit 6 aktif edilmişse konuşmacı olarak ayarlanır. Bit 7 asla kullanılmayan bir önemsiz (don't

care) bitidir ve “0” olarak kabul edilir. Bir cihaz için 0x05 adresli komut mesaj baytını dinleyici olarak ayarlamak için 0x25 olmalı, konuşmacı olarak ayarlamak için 0x45 olmalıdır (Park and Mackay 2003).

### 2.2.9. GPIB adres kaldırma

İki veya daha fazla cihaz arasındaki iletişimde kontrolör yeni bir konuşmacı ve dinleyici atamadan önce o anki konuşmacı ve dinleyicinin yolunu temizlemelidir. Kullanılmayan bir konuşma adresi gönderme ya da daha çok un-talk (UNT) komut mesajı (0x5F) gönderme o anki konuşmacıyı susturur. Bu komut çokta gerekli değildir. Çünkü bir konuşmacıyı aktif yapmak zaten diğer tüm konuşmacıları pasif yapar. O anki dinleyici ya da dinleyiciler un-listen (UNL) komut mesajı (0x3F) ile otomatik olarak pasif edilir. Daha öncesinde birden fazla dinleyici adreslenmişse tek bir dinleyici pasif edilemez (Park and Mackay 2003).

### 2.2.10. Veri mesajlarını sonlandırma

Veri mesajlarını sonlandırmanın 3 yolu vardır.

- EOI yöntemi – Konuşmacı veri yolundaki verinin en son baytına EOI eklediğinde EOI aktif hale gelir ve bu hat tüm dinleyiciler tarafından görülür. Bu şekilde mesaj sonlandırılmış olur.

- EOS yöntemi – Bu yöntem kontrol yol hatlarını kullanmaz. Bunun yerine daha önceden belirlenmiş bir end-of-string baytı veri mesajının sonuna eklenir ve bu veri baytı dinleyici veya dinleyicilere gönderilir. EOS baytı genelde yeni satır karakteri (0x0A) veya satır başı karakteri (0x0D)’dir.

- Count Yöntemi – Bu yöntemin uygulanabilmesi için belirlenmiş sayıdaki mesaj baytları dinleyiciye ulaşır ulaşmaz kontrolör konuşmacının daha fazla mesaj baytı

göndermesini kesmesi gerekir. NRFD ve NDAC yol kontrol hatlarının aktif edilmesi ile bu gerçekleştirilebilir. Fakat, bunun için GPIB donanımına uygulama yazılımının yüklenmesi gerekir (Park and Mackay 2003).

### 2.2.11. Veri gönderme ve alma

GPIB'in doğru bir şekilde ayarlanıp başlatıldığını varsayarsak, veri gönderme ve alma protokolleri aşağıdaki gibi olur:

Veri gönderme:

1. ATN'yi "true" ve EOI "false" yap
2. Kontrolör konuşma adresini gönder
3. UNL komut mesajını gönder
4. Dinleyici olarak ayarlanmış her bir cihaz için dinleme adresini gönder
5. ATN'yi "false" yap
6. Gerekli veri baytlarını aynı anda bir bayt olacak şekilde gönder
7. Sonlandırma karakteri gönder ya da EOI'yı "true" yap.

Veri Alma:

1. ATN'yi "true" ve EOI'yı "false" yap
2. UNL komut mesajını gönder
3. Kontrolör dinleme adresini gönder
4. Gerekli konuşmacı adresini gönder
5. ATN'yi "false" yap
6. Sonlandırma karakteri alana kadar ya da EOI "true" olana kadar veri baytlarını al (Park and Mackay 2003).

### 2.2.12. IEEE 488.2

IEEE 488.2, orijinal IEEE-488 standardında ortaya çıkan sorunları düzeltmek için tasarlandı. Aynı zamanda mevcut IEEE-488.1 standardındaki cihazlar ile uyumu garanti etmek için tasarlandı. Standardı genişletmekte kullanılan yöntemde kontrolörler ve cihazların konuşmacı gibi görev yapabileceği net bir şekilde tasarlanmalıydı. Bunun anlamı IEEE-488.2 ile uyumlu sistemler güvenilir ve verimli olmalıydı. Ancak IEEE-488.1 cihazlarıyla geriye doğru uyumluluk, tasarlanan standart tarafından sağlanabilirdi. IEEE 488.2 cihazları dinleyici olarak çalıştıklarında, IEEE-488.1 cihazlarından gelen geniş çapta veri formatları ve komutları kabul edilebilirdi (Park and Mackay 2003).

#### 2.2.12.a. IEEE-488.2 kontrolörlerinin gereksinimleri

IEEE-488.2 standardı kontrolör için gerekli IEEE-488.1 arayüz kabiliyetlerini içeren bir grup ihtiyaçları aşağıdaki gibi düzenlemiştir.

- SRQ hattının geçiş ve durumunu algılama
- EOI'yı sezme ve ayarlama
- 100 mikrosaniye için arayüz temizleme hattını darbeleme
- REN hattını ayarlama ve sürme (aktif/pasif)
- Herhangi giriş çıkış işlemi için zaman aşımı

#### 2.2.12.b. IEEE-488.2 kontrol sıralaması

Eklenen özelliklerden biri kontrolör tarafından gönderilen tam mesajların tanımlanmasının yanısıra gönderilen mesajlar birden fazla ise bu mesajların sıralanmasıdır. Çizelge 2.4'te IEEE-488.2 on beş gerekli kontrol sıralaması ve dört opsiyonlu kontrol sıralaması tanımları gösterilmektedir. Bu kontrol sıralamaları cihazların özel mesajlara nasıl cevap vereceğini ve tanımlanan her bir işlemin komut

mesajlarının nasıl sıralanacağını göstererek GPIB'in tam durumunu tanımlar (Park and Mackay 2003).

**Çizelge 2.4.** Gerekli ve opsiyonel kontrol sıralaması (Park and Mackay 2003)

| Description | Control sequence | Compliance |
|---|---|---|
| Send ATN-'true' commands | Send command | Mandatory |
| Set address to send data | Send setup | Mandatory |
| Send ATN-'false' data | Send data bytes | Mandatory |
| Send a program message | Send | Mandatory |
| Set address to receive data | Receive setup | Mandatory |
| Receive ATN-'false' data | Receive response message | Mandatory |
| Receive a response message | Receive | Mandatory |
| Pulse IFC line | Send IFC | Mandatory |
| Place devices in DCAS | Device clear | Mandatory |
| Place devices in local state | Enable local controls | Mandatory |
| Place details in remote state | Enable remote | Mandatory |
| Place devices in remote with local lookout state | Set RWLS | Mandatory |
| Place devices on local lockout | Send LLO | Mandatory |
| Read IEEE 488.1 status byte | Read status byte | Mandatory |
| Send group execution trigger (GET) message | Trigger | Mandatory |
| Give control to another device | Pass control | Optional |
| Conduct a parallel poll | Perform parallel poll | Optional |
| Configure device's parallel poll responses | Parallel poll configure | Optional |
| Disable device's parallel poll capability | Parallel poll unconfigure | Optional |

### 2.2.12.c. IEEE-488.2 protokolleri

Protokoller ortak test fonksiyonlarını gerçekleştirmek için birkaç kontrol sıralamasını birleştiren yüksek seviye fonksiyonlardır. Protokoller sayesinde test programı geliştirme zamanı azalır. IEEE-488.2 Çizelge 2.5'te gösterilen iki zorunlu ve 6 opsiyonel protokole sahiptir.

**Çizelge 2.5.** IEEE-488.2 Kontrolör Protokolleri (Park and Mackay 2003)

| Keyword | Name | Compliance |
|---|---|---|
| RESET | Reset system | Mandatory |
| FINDRQS | Find device requesting service | Optional |
| ALLSPOLL | Serial poll all devices | Mandatory |
| PASSCTL | Pass control | Optional |
| REQUESTCTL | Request control | Optional |
| FINDLSTN | Find listeners | Optional |
| SETADD | Set address | Optional but requires FINDLSTN |
| TFSTSYS | Self-test system | Optional |

Fonksiyonlar tanımlanan protokollerin her biri tarafından aşağıdaki gibi gerçekleştirilir:
RESET: GPIB yolunu başlatır, temizler ve tüm cihazları bilinen bir duruma ayarlar.
FINDRQS: Kontrolör SRQ hattının "False"dan "True"ya geçişini algılar ve ilk olarak en kritik cihazı onarır.
ALLSPOLL: Herbir cihaz seri bir şekilde sorgulanır ve her bir cihazın durum baytını geri döndürür.
PASSCTL: Yol kontrolünü diğer cihazlara geçirir.
REQUESTCTL:Yol kontrolü ister.
FINDLSTN: Kontrolör bu komutu belirli bir dinleyicinin adresiyle yayınlar ve NDAC handshake hattını o adreste bir cihaz olup olmadığına karar vermek için izler.
SETADD: FINDRQS ile bir cihaz adresi ayarlamada kullanılır.
TESTSYS: Kontrolör her bir cihaza kendi kendini test etmesi ve herhangi bir problem olup olmadığını geri raporlaması için emir verir.

Protokollerin kullanışlılığının bir örneği olarak GPIB sistemindeki bir veya daha fazla cihazın seri olarak sorgulanması düşünülebilir. Birçok cihaz SRQ hattını aktif ederek kontrolörden eş zamanlı olmadan servis isteğinde bulunabilir. Bilinen aktif cihazları sorgulayarak hangi cihaz(lar)ın servis isteğinde bulunduğuna karar vermek kontrolörün görevidir. Belirli bir cihazın seri sorgulaması IEEE-488.1 protokolü ile gerçekleştirilebilir.

Aşağıdaki olaylar dizisi meydana gelebilir:

- Kontrolör un-listen (UNL) komut mesajı göndererek tüm dinleyicilerin dinlemesini pasif yapar.
- Kontrolörün dinleme adresi ayarlanır.
- Kontrolör SPE (seri sorgulama aktif-serial poll enable) komutunu içeren komut mesajını cihaza gönderir. Bu komut cihazdaki IEEE-488.1 seri sorgu modunu ayarlayarak cihazın seri sorgu durum baytını geri göndermesini sağlar. Cihaz konuşmaya ayarlandığında seri sorgu durum baytını geri gönderir.
- Kontrolör geçici olarak gereken cihazı, yola cihazın konuşma adresini koyarak konuşmacı olarak ayarlar.
- Kontrolör konuşmacı olarak ayarlanan cihazdan durum baytını okur.
- Kontrolör SPD (serial poll disable- seri sorgulama pasif) komutunu içeren komut mesajını cihaza gönderir. Bu komut cihazdaki seri sorgu modunu resetler.

Zorunlu ALLSPOLL protokolu GPIB'deki adresleri protokol tarafından sağlanan (Örneğin ALLSPOLL 1,2,3,4) tüm cihazları seri olarak sorgular. Her bir cihazda seri sorgulama gerektiren bu adımlar kullanıcıya görünmeden yürütülür (Park and Mackay 2003).

### 2.2.12.d. Cihaz arayüz kapasiteleri

IEEE-488.2 bir cihazın sahip olması gereken IEEE-488.1 arayüz özelliklerinin minumum grubunu tanımlar. Bu cihaz arayüz özellikleri her bir cihazın altında veya yakınında bulunması gereken açık kodlar tarafından gösterilir. Çizelge 2.6'da bir cihazın minumum gerçekleştirmesi gereken fonksiyonlar gösterilmiştir.

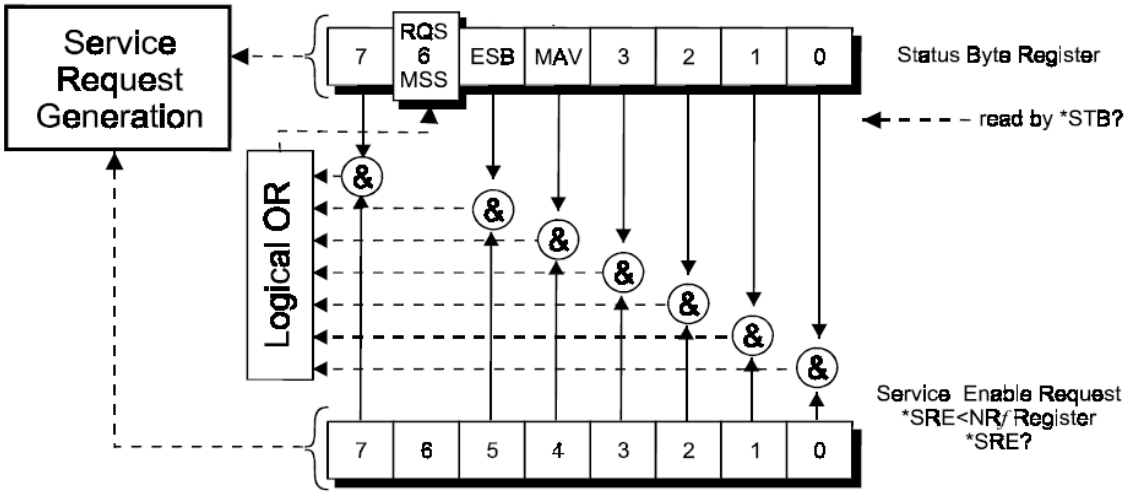
**Çizelge 2.6.** IEEE-488.2 için cihaz arayüz kapasitesinin minumum serisi (Park and Mackay 2003)

| Capability | Code | Comment |
|---|---|---|
| Source handshake | SH1 | Full capability |
| Acceptor handshake | AH1 | Full capability |
| Talker | T(TE)5 or T(TE)6 | Basic talker, serial poll or untalk on MLA |
| Listener | L(LE)3 or L(LE)4 | Unlisten on MTA or full capability |
| Service request | SR1 | Full capability |
| Device clear | DC1 | Full capability |
| Remote local | RL0 or RL1 | No or full capability |
| Parallel poll | PP0 or PP1 | No or full capability |
| Device trigger | DT0 or DT1 | No or full capability |
| Controller | C0 or C4 with C5, C7, C8 or C11 | No or full capability with send if message, pass, receive control |
| Electrical interface | E1 or E2 | Open collector or tri-state |

Kullanıcı her kodun neyi temsil ettiğininin tam bir açıklamasını ve cihaz işlevlerini açıklayan diğer tüm kodları IEEE-488 standardı ya da alternatif olarak belirli bir cihazın özellikleri olarak ifade etmelidir (Park and Mackay 2003).

### 2.2.12.e. Durum raporlama modeli

IEEE-488.1 ile ilgili problemlerden biri de birbirinden ayrı cihazlardan durum raporlama yapmasıydı. Bu, her bir cihazın durum baytındaki farklı bitleri kullanarak durum bilgilerini raporlaması şeklindeydi. IEEE-488.2 Şekil 2.13'te gösterildiği gibi bu problemi standart bir durum raporlama modeli tanımlayarak çözdü (Park and Mackay 2003).




**Şekil 2.12.** IEEE-488.2 Durum raporlama modeli (Park and Mackay 2003)

Bu model sadece RQS (request service) biti olarak tanımlı bit 6'ya sahip olan IEEE-488.1'in durum bayt tanımlasının üzerine inşa edilmiştir. RQS bit bir cihazın RQS hattını ileri sürerek (aktif yapma) servis isteğini göstermeye yarar. Öteki tanımlanan iki bit ESB (event status bit-olay durum biti) ve MAV (message available-mesaj var)'dır. Diğer bitler üretici tarafından tanımlanır. ESB biti standart olay durum kaydedicisinde tanımlanan olaylardan birinin meydana geldiğini gösterir. Sadece kullanıcının standart olay durum kaydedicisinde yerini tutan bit'i ayarladığı olaylar durum baytındaki ESB bitinin ayarlanmasına sebep olacaktır. MAV bit'i cihazın çıkış kuyruğundaki mesajın hazır olduğunu göstermek için kullanılır.

Eğer bir cihaz durum kaydedicisindeki bit'lerden birini aktif veya pasif ayarlandığında SRQ hattını ileri sürerse, servis request enable kaydedicisi kullanılır. Eğer bu kaydedicideki bir bit kullanıcı tarafından ayarlanırsa, o zaman bu kaydedicide yerini tutan bitin ayarlanması cihazın SRQ hattını ileri sürmesine sebep olacaktır (Park and Mackay 2003).

### 2.2.12.f. Ortak komut seti

IEEE-488.2 cihazın sahip olması gereken minumum bir standard komut setine de sahiptir. Bunlar yeni yol komut mesajları değildir, ancak tüm cihazlarda ortak yeni veri mesajlarıdır. Çizelge 2.7'de her bir komutun hangi komut grubunda olduğunu gösteren zorunlu IEEE-488.2 komutları listelenmiştir.

**Çizelge 2.7.** IEEE-488 Komutları (Park and Mackay 2003)

| Mnemonic | Group | Description |
|---|---|---|
| *IDN? | System data | Identification query |
| *RST | Internal operations | Reset |
| *TST? | Internal operations | Self-test query |
| *OPC | Synchronization | Operation complete |
| *OPC? | Synchronization | Operation complete query |
| *WAI | Synchronization | Wait to complete |
| *CLS | Status and event | Clear status |
| *ESE | Status and event | Event status enable |
| *ESE? | Status and event | Event status enable query |
| *ESR? | Status and event | Event status register query |
| *SRE | Status and event | Service request enable |
| *SRE? | Status and event | Service request enable query |
| * STB? | Status and event | Read status byte query |

SCPI (Programlanabilir cihazlar için standart komutlar) farklı üreticilerden her bir cihaz sınıfı (yani multimetre, osilaskop, dijital voltmetre v.b.) için özel komutlar tanımlayarak daha fazla ortak komut seti elde eder.

### 2.2.12.g. Programlanabilir cihazlar için standart komutlar (SCPI-Standart commands for programmable instruments)

Farklı üreticiler tarafından geliştirilen programlama cihazları için komut setleri arasındaki standartlaştırmanın yetersizliği bir grup üreticinin SCPI özelliklerini geliştirmesine neden oldu.

Örneğin, 'MEAS: FREK? ' ya da ': MEAS: VOLT' komutu sırasıyla frekans ve voltaj ölçümü okumak için, bu parametrelerin işleme kapasitesine sahip herhangi bir cihaz için uygulanabilir. Cihaz, bir voltmetre, osiloskop, ya da bir frekans sayacı olabilir.

IEEE 488.1, IEEE 488.2 ve SCPI cihazları ve kontrolörleri aynı sistem içinde kullanılabilir ama en kolay programlanabilir, esnek, hızlı ve değiştirilebilir sistemler IEEE 488.2 kontrolörleri ve SCPI cihazlarını kullanır (Park and Mackay 2003).

### 2.2.12.h. SCPI için gerekli olan IEEE-488.2 ortak komutları

SCPI için minimum gereksinim olarak, IEEE 488.2 standardında tanımlanmış ve tüm SCPI uyumlu cihazların ihtiyaç duyduğu ortak ve zorunlu komutlar Çizelge 2.7'de tanımlanmıştır. Bu minimum komut seti cihaz özel komutlarını ele almamıştır, fakat cihaz resetleme, cihazın kendini testi, servis isteği etkinleştirme raporlaması, cihaz kimlik tanımlama, işlev eş zamanlaması ve standard olay durum etkinleştirmesi ve raporlaması gibi durum sorguları ve program komutlarından oluşur (Park and Mackay 2003).

### 2.2.12.i. SCPI için gerekli komutlar

IEEE-488.2'de tanımlanmış ortak komut setlerini oluşturmak için, Çizelge 2.8'de gösterildiği gibi SCPI, kendi gerekli ortak komut setini de tanımlar.

Örneğin, IEEE-488.2 tarafından tanımlanmış durum raporlama OPERation ve QUEStionable durum kaydedicileriyle genişletilmiştir. Genişletilmiş şeklinde içerikleri okumak için EVENt ve CONDition kaydedicileri, maskelemeyi ayarlamak ve okumak için ENABle maske komutları vardır.

SYSTem üst komutunun girişi tarafından izin verilen komut seti TIME ya da SECurity erişim ayarları gibi genel bakım görevlerini gerçekleştiren fonksiyonları tanımlar. Alt komut ERRor? sorgusu SCPI cihazının hata/olay kuyruğundan bir sonraki girişi ister.

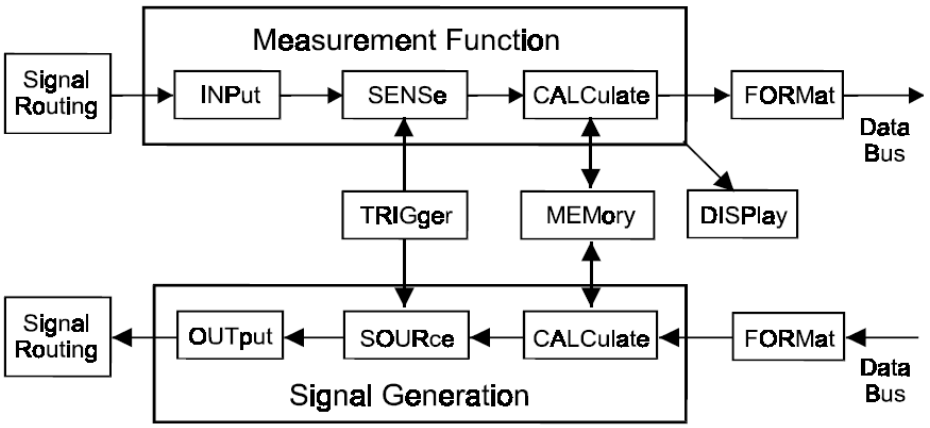
**Çizelge 2.8.** SCPI için Gerekli Komutlar (Park and Mackay 2003)

| Command | Description |
|---|---|
| :SYSTem | Collects functions not related to instrument performance |
| :ERRor? | Requests the next entry from the instruments queue |
| :STATus | Controls the SCPI-defined status reporting structures |
| :OPERation | Selects the operation structure |
| [:EVENt]? | Returns the contents of the event register |
| :CONDition? | Returns the contents of the condition register |
| :ENABle? | Reads the enable mask |
| :QUEStionable | Selects the questionable structure |
| [:EVENt]? | Returns the contents of the event register |
| :CONDition? | Returns the contents of the condition register |
| :ENABle | Sets the enable mask which allows event reporting |
| :ENABle? | Reads the enable mask |
| :PRESet | Enables all required event reporting |

### 2.2.12.j. SCPI programlama komut modeli

Komutları gruplara ayırma ve uyumluluğun elde edilmesiyle, Şekil 2.14'te gösterildiği gibi SCPI, programlanabilir bir cihazın tüm farklı tip cihazlara uygulanabilen bir modelini tanımlar.

Bu modeli kullanırken bir cihazın genelleştirilmiş modeli olduğu ve bunun tüm cihazlarda işlevsellik göstermeyeceği bilinmelidir. Örneğin, bir osiloskop SCPI modelindeki sinyal üretim bloğu tarafından tanımlanmış işlevselliğine sahip değilken, fonksiyon jeneratörü sahip olacaktır. Buna ek olarak fonksiyon jeneratörü de muhtemelen ölçüm fonksiyon bloğu tarafından tanımlanan işlevselliğe sahip olmayacaktır (Park and Mackay 2003).




**Şekil 2.13.** SCPI cihaz modeli (Park and Mackay 2003)

SCPI cihaz modelinin fonksiyonel bileşenleri ilerleyen bölümlerde tanımlanmıştır.

**- Ölçüm fonksiyonu**

Ölçüm fonksiyon bileşenleri bir sinyali önişlenmiş şekle dönüştürmek için kullanılır. Bu üç farklı bölüme ayrılır.

**INPut**: Gelen işlenmemiş sinyalde filtreleme, yönlendirme ve zayıflatma gibi sinyali forma sokma işlemlerini gerçekleştirir.
**SENse:** Forma sokulmuş giriş sinyalini kullanıcının işleyebileceği veri formatına dönüştürür. Bu fonksiyon alan, çözünürlük, geçit zamanı ve norml mod reddetme gibi parametreleri kontrol eder.
**CALCulate**: Veri formatını belirli bir uygulama için daha kullanışlı bir formata çevirir.

Ölçüm fonksiyonu cihazın işlevselliği değil de sinyal parametreleri tarafından belirlenen yukarıdaki üç fonksiyon tarafından belirlenir. Bu, genellikle aynı ölçümleri SCPI komutunu bozmadan değiştirilebilen farklı cihazlara izin vererek en yüksek seviyede uyumluluk sağlar (Park and Mackay 2003).

**- Sinyal üretimi**

Sinyal üretim fonksiyonu dahili veriyi gerçek sinyallere dönüştürür. Bu da üç farklı kısıma bölünür:

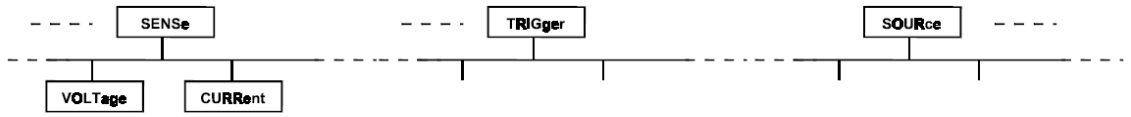
**OUTput:** Bu sinyal bloğu çıkış sinyalini filtreleyerek, zayıflatarak ve yönlendirerek şekillendirir
**SOURce:** Akım, gerilim, güç ve frekans gibi sinyal parametrelerini belirterek dahili veri ve özel karakteristiklere dayalı bir sinyal üretir.
**CALCulate:** Bu fonksiyonel blok veriyi bir mühendislik biriminden diğerine çevirir ve sinyal verisinin üretiminde ortaya çıkacak anormallikleri de hesaba katar.

**- Sinyal bozma**

Sinyal yönlendirme fonksiyonu belirli bir cihazın iç fonksiyonlarına giden sinyalin bağlantısını kontrol eder. Trigger bileşeni analog veya dijital sinyal gibi cihazın dış şartları ile cihazın işlevselliğini ve yazılım komutunu içeren iç cihaz olaylarını senkronize eder. Format bileşeni cihazdan veriyi -standart yol üzerinden iletimi için - uygun bir forma dönüştürürken hafıza bileşeni verinin dahili depolanması için kullanılır. DISPlay bileşeni verinin ekranda görünmesine izin verir (Park and Mackay 2003).

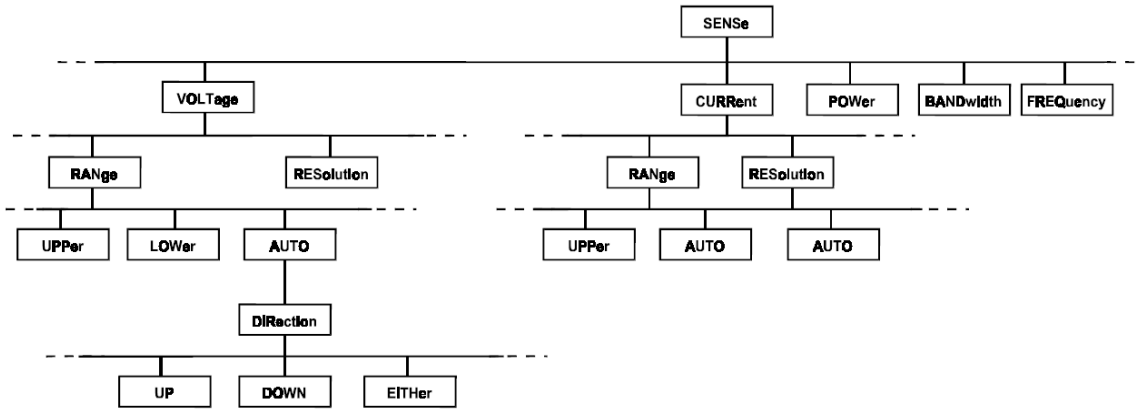
### 2.2.12.k. SCPI hiyerarşik komut yapısı

SCPI cihaz modeli her bir ana komut için fonksiyonel blok tanımlar. Bu kategorilerin her biri hiyerarşik bir komut yapısına sahip olup komut ağacı olarak adlandırılır. Bu komut ağacı alt komutları, parametreleri ve gerçekleşek olan fonksiyonun belirli detaylarını sağlayan seçenekleri içerir.




**Şekil 2.14.** SCPI hiyerarşik komut yapısı (Park and Mackay 2003)

SENSe kısmi komut ağacı bir sinyali gereken dahili veri formatına dönüştürmeyi kontrol eden bir cihazı programlayan alt programları içerir. Bu veri, kullanıcı tarafından daha fazla değiştirilebilir.




**Şekil 2.15.** SENse komut yapısı için kısmi komut ağacı (Park and Mackay 2003)

SENSe komutları çözünürlük ve aralık gibi parametreleri kontrol eder. Bu yüzden bir dijital voltmetreyi otomatik aralıkta ölçüm yapmak için programlamak aşağıdaki gibi bir komutla olacaktır.

SENS:VOLT:RANG:AUTO:DIR:EITH

İki nokta üst üste her bir komutu ayırt etmek ve cihaz komut yorumlayıcısını komut ağaç hiyerarşisindeki bir sonraki seviyeye taşımak için kullanılır. Komut kelimeleri herhangi bir boşluk ile ayrılmaz.

Bir dijital voltmetreyi 5 ve 10 gibi belirli alt ve üst değerleri ayarlayarak ölçmeye programlamak için gerekli komutlar aşağıdaki gibidir.

SENS:VOLT:RANG:UPP10:LOW 5

Bu komutta hiyerarşik seviyesi aynı olan iki alt komut vardır. İki komut seviyeleri değişmeden yayınlandığında komutları ayırmak için noktalı virgül kullanılır. Komutlar ve herhangi ilişkili parametreleri boşluk ile ayrılmalıdır (Park and Mackay 2003).

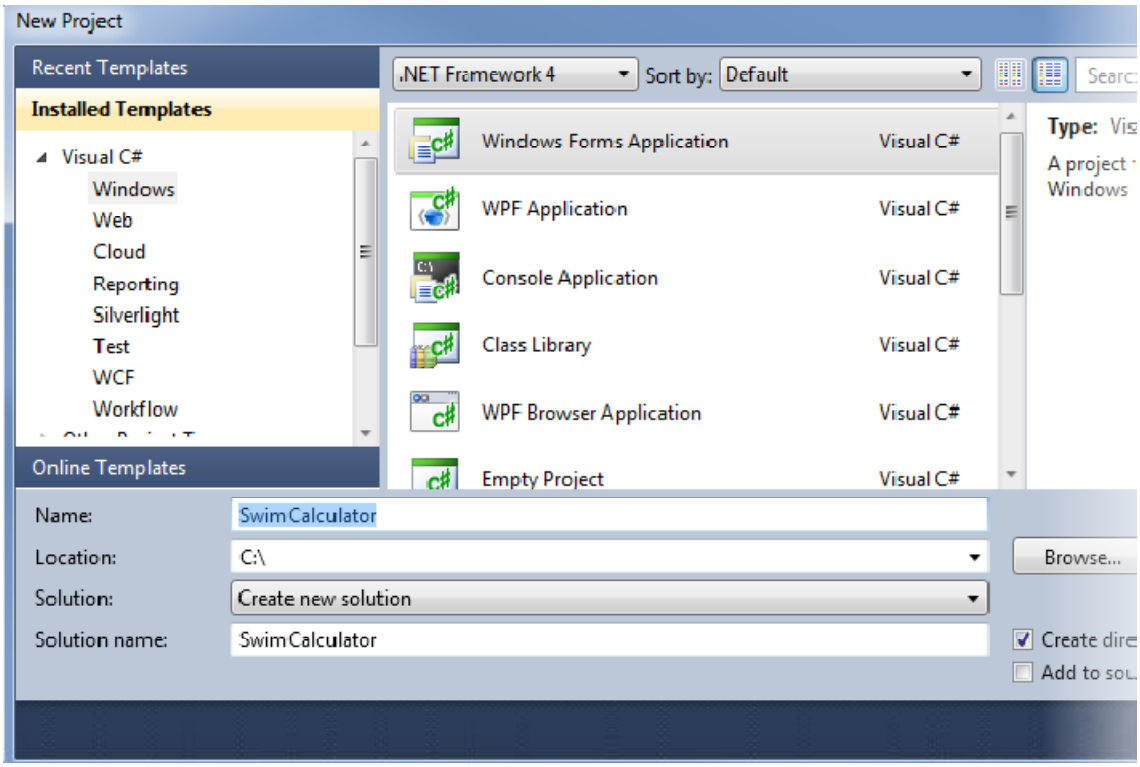
## 2.3. VISUAL C#

### 2.3.1. C# tanıtımı

C# bir programlama dilidir. Her dilde olduğu gibi sözlük ve söz dizimine sahiptir. C# veya Java gibi programlama dilleri İngilizce ve Fransızca gibi doğal dillerden farklıdır. Yeni bir programlama dili öğrenmek doğal bir dil öğrenmekten daha kolaydır. Bir programlama dili sözlüğünde sadece bir miktar kelime vardır; C# yaklaşık 100 anahtar kelimeye sahiptir. Bu fazla görünebilir ancak bazı anahtar kelimeler diğerlerinden daha fazla kullanılır ve muhtemelen bazıları hiç kullanılmaz. Programlama dilleri belirli bir şekilde kullanılması gereken anahtar kelimeler olan çok kesin söz dizimlerine sahiptir. İngilizce çok esnek ve akıcıdır. Programlama dilleri komutları bilgisayara mümkün olduğunca açık ve net bir şekilde ifade etmek için kullanılır. Bu yüzden programlama dilleri kısa ve kesindir. İşlevsel olarak C# için bir genel, otoriter tanım vardır. Gramer uzmanları İngilizce'deki kelimelerin anlamını ve tarihini tartışabilirler fakat C#'da ki kelimelerin tarihi ve anlamı Microsoft tarafından tanımlanmıştır. Bunlar "http://msdn.microsoft.com/en-us/library/ms228593.aspx" adresinde bulunabilir (Freeman 2010).

### 2.3.2. Proje oluşturma

İlk adım Windows Form Proje'si oluşturmaktır. Şekil 2.17'de gösterildiği gibi Visual Studio File Menü'sünden New > Project seçilir ve açılan template listesinden Windows Forms Applicatin seçilir. Mevcut template seti, yüklenen Visual Studio'nun sürümüne ve yüklenirken hangi seçeneklerin seçildiğine bağlıdır (Freeman 2010).




**Şekil 2.16.** Windows Forms Application template seçme (Freeman 2010)

Projeye bir isim verilir ve OK butonu tıklanır. Yeni proje iki anahtar dosya içerecektir. Birincisi Program.cs'dir. Program.cs program için ana methodları içeren kod dosyasıdır. Program.cs'nin içeriği aşağıdaki gibi olacaktır.

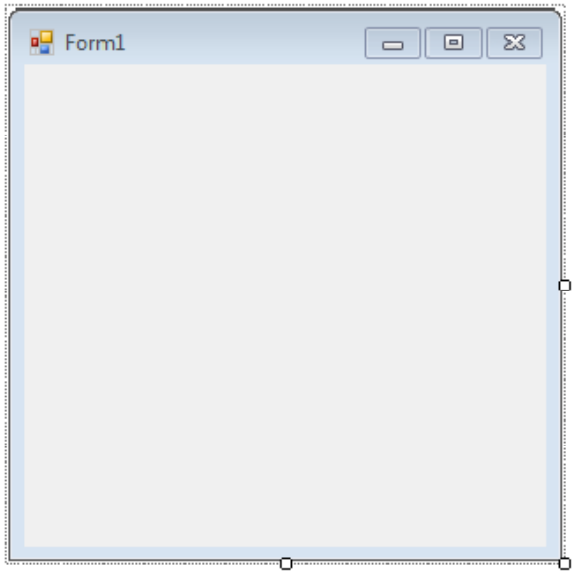
```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Windows.Forms;
namespace SwimCalc
{
        static class Program
        {
                static void Main()
                {
                        Application.EnableVisualStyles();
                        Application.SetCompatibleTextRenderingDefault(false);
                        Application.Run(new Form1());
                }
        }
}
```

Bu dosyayı düzenlemeye gerek yoktur fakat Windows Forms programının herhangi bir diğer C# programı gibi bir sınıf ve bir ana metoda sahip olduğu görülebilir. Ana metottaki kod ifadesi Windows Forms sınıflarının birçoğunu içeren System. Windows. Forms uzayındaki Application sınıfının üyelerini çağırır. Buradaki en önemli ifade şudur;

```
Application.Run(new Form1());
```

Bu ifade yeni bir Form1 nesnesi oluşturur ve onu Application.Run metoduna gönderir. Form1 sınıfı projedeki Form1.cs kod dosyasında yer alır. Eğer Solution Explorer'daki bu dosya çift tıklanırsa, Windows Forms tasarım yüzü görünecektir. Şekil 2.18'deki tasarım yüzü program için ana Windows Form'udur.

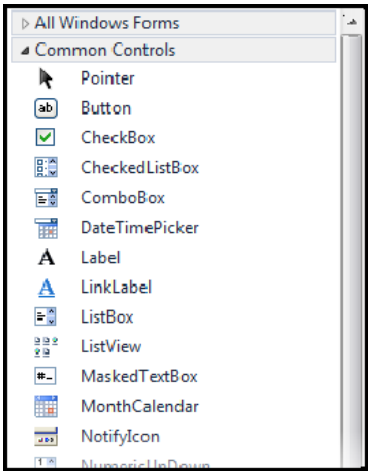



**Şekil 2.17.** Boş Tasarım Yüzü (Freeman 2010)

Tüm kullanıcı arayüz teknolojileri tasarım yüzeyinde kontrolleri istenilen yere sürüklenip bırakılabilecek ve onları görünmesi istenildiği gibi ayarlayabilecek görsel tasarımcılara sahiptir. Kontroller bir kullanıcı arayüz elementleridir. Temel öğelerden (button, label gibi) karışık bileşenlere (data grids ve data pickers gibi) kadar değişir ve program fonksiyonlarıyla yardım edebilecek öğeleri içerir fakat bunlar (data sources ve LINQ queries) kullanıcıya görünmezdir. Şekil 2.18'deki tasarım yüzü boş gri bir penceredir. Çünkü henüz herhangi bir şey yerleştirilmedi. Herhangi bir tasarım çalışması yapılmasa bile bazı bağımsız fonksiyonlar vardır. Eğer proje derlenir ve çalıştırılırsa aynı boş pencere görünecektir. Görev çubuğundaki ikon ve pencerenin sağ üst tarafındaki buton gerektiği gibi çalışır (Freeman 2010).

### 2.3.3. Kontrol (Control) ekleme

Windows Forms Kontrolleri'nin listesi Visual Studio View menüsünden açılabilen Toolbox penceresinde mevcuttur. Kontroller Şekil 2.19'da gösterildiği gibi common controls, containers, data ve vb. şekilde birlikte gruplandırılmıştır.




**Şekil 2.18.** Windows Forms Kontroller Araç Çubuğu (Toolbox) (Freeman 2010)

Çok sayıda bileşenin olduğu Toolbox (araç çubuğu)'tan görülebilir fakat standart sette olmayan ya da standart kontrollerin daha fazla fonksiyonel versiyonları olan üçüncü kişi kontrolü için de bir aktif market vardır. Eğer standart kontroller kullanılarak istenilen tasarım yapılamazsa, farklı fiyat ve kalitede olmak üzere, mevcut olan çok etkili kontrol kütüphane eklentileri kullanılabilir (Freeman 2010).

Standart kontroller kullanılarak örnek bir program yapılabilir. Programdaki tasarım yüzüne bir kontrol eklemek için toolbox penceresinden eklenmek istenen kontrol öğesi tasarım yüzüne sürüklenir. Toolbox penceresindeki bazı kontroller diğerlerinin düzenlenmesine yardımcı olur. Bu kontroller, kullanıcının pencereyi tekrar boyutlandırmasında arayüzü uygun bir şekilde düzenlemesinden ziyade karmaşık programları tasarlanmasında yardımcı olur. Bu örnekte kesin yerleştirme (absolute positioning) kullanılacaktır. Yani tasarım penceresi tekrar boyutlandırıldığında eklenen kontrolün yeri değişmez. Örnek olarak yeri önemsenmeden bir buton kontrolü tasarım yüzeyine sürüklenirse projeye bir buton yer tutucunun eklendiği gözlemlenecektir. Solution Explorer penceresi kullanılarak Form1.cs öğesi genişletilip Form1.Designer.cs kod dosyası çift tıklanır. Eğer *Windows Form Designer generated code* yazan kısmın sol kenarındaki hatta bulunan artı işareti tıklanırsa buton kontrolü eklendiğinde

programa hangi kodların eklendiği görünür. Şu şekilde bazı ifadeler olacaktır (Freeman 2010).

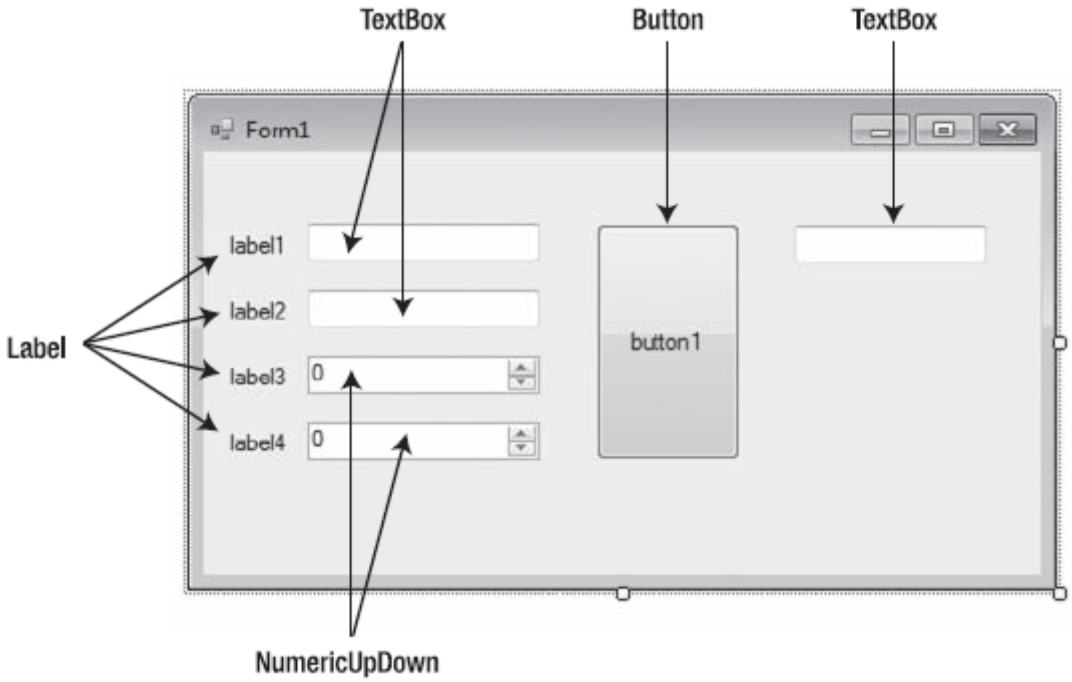
```
this.button1.Location = new System.Drawing.Point(83, 91);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(75, 23);
this.button1.TabIndex = 0;
this.button1.Text = "button1";
this.button1.UseVisualStyleBackColor = true;
```

Bunlar buton tasarım yüzüne sürüklendiğinde otomatik olarak üretilmiştir. Tasarım penceresine git ve buton kontrolünü sol üst köşeye sürükleyerek taşı. Eğer Form1.Designer.cs sekmesine geri gidilirse, ilk ifadenin şuna benzer şekilde değiştiği görülür (Freeman 2010).

```
this.button1.Location = new System.Drawing.Point(3, 3);
```

*Windows Form designer* tasarım yüzüne eklenenlere bağlı olarak bizim için C# kodu üretir. Buton hareket ettirildiğinde üretilen kod değişimi yansıtacak şekilde güncellenir. Eğer buton silinirse, onun için üretilen tüm kodlar silinecektir. Benzer şekilde tutunacak yerlerin bir veya daha fazlasının sürüklenmesiyle boyutları değiştirilebilir ve bir veya daha fazla kontrol seçilip *Visual Studio Format* menüsü kullanılarak hizalanabilir veya dağıtılabilir. *Alignment, spacing* v.b. işlemler için menü öğeleri vardır. Tasarım yüzündeki ana pencere de ana program penceresinin başlangıç boyutlarının değiştirilmesinde kullanılabilen tutunacak yerlere sahiptir. *Windows Forms Designer'ı* kullanmaya hiç gerek yoktur. C# ifadelerini kendi kendisine oluşturulabilirsiniz fakat bu tekrarlanan ve hataya müsait bir yaklaşımdır. Projeleriniz için tasarımcının kullanılması önerilir (Freeman 2010).

Projede kontroller tasarım yüzüne sürüklenir ve Şekil 2.20'dekine benzer bir şekilde düzenlenir ve tekrar boyutlandırılır. Her bir kontrolün tipi şekilde gösterilmiştir. Alt ve üst kısımda bir geniş boşluk bırakıldığından emin olunur. Bu boşluğa daha sonra menu ve *status bar* eklenecektir. Kontroller hareket ettirildiğinde ve terkar boyutlandırıldıklarında kontolleri düzgünce hizalamak için diğer kontrollere ait boyut ve pozisyonları gösteren kılavuz çizgileri görünür (Freeman 2010).




**Şekil 2.19.** Kontrollerin tasarım yüzüne sürüklenmesi (Freeman 2010)

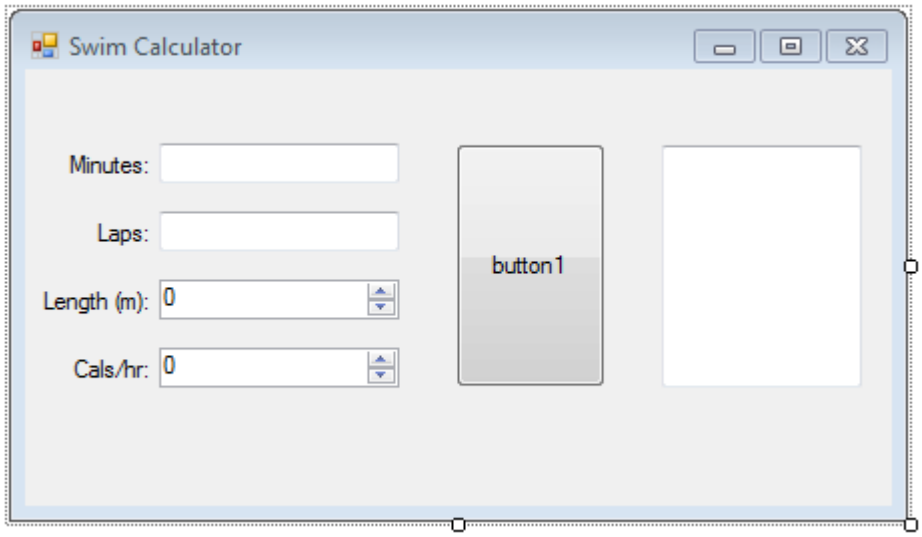
### 2.3.4. Kontrol özelliklerini ayarlama

Tasarım yüzündeki her bir kontrol bir grup özelliğe sahiptir. Bu özellikler düzen ve görünümü belirler. Üretilen C# koduna bakıldığında görülen ifadelerin her biri bir kontrol özelliği için daha önce ayarlanmıştır. Bu bölümde kontrollerin özellikleri değiştirilecektir. Böylece onlar istenilen şekilde görünür ve davranır. Formun kendisi seçilerek başlanır. Bu pencerenin üstü tıklanılarak yapılabilir. Properties pencerisi açılır

ve özellikler listesi görülür. Pencere Text özelliğini görünene kadar aşağı kaydırılır. Bu Text özelliğinde görülen isim "Swim Calculator" olarak değiştirilir. Tasarım yüzündeki pencere adının değiştiğine dikkat edin. Daha karmaşık bir örnek yapalım. Tasarım yüzündeki ekranın sağ tarafındaki TextBox'ı seçiniz. Properties penceresinde *Multiline* özelliğini bul ve değeri "True" olarak değiştir. Kontrol yer tutucusu üzerinde ilaveten, kontrolü tekrar boyutlandırmaya izin veren, tutunacak yerler görülecektir. Varsayılan olarak TextBox kontrolü tek satır *text* gösterir fakat *Multiline* özelliği aktif yapılırsa çoklu satır gösterebilir. Tutacak yerleri kullanarak TextBox'i yanındaki butonla aynı yüksekliğe getirebilirsiniz (Freeman 2010).

### 2.3.5. Etiketleri (Label) düzenleme

Label kontrolü için değiştirilmesi gereken özellik sadece Text özelliğidir. Bu pencerede label kontrolü ne göstereceği ayarlanır. En üstteki Label'dan başlayarak Text özelliklerini "Minutes:", "Laps:", "Length (m):", ve "Cals/hr:" olarak değiştirilir. Tasarım yüzündeki her bir label'ı sıra ile seçiniz, Text özelliğini bulunuz ve gereken değeri ayarlayınız. Her şeyin uygun olduğundan emin olmak için kontrollerin yerlerinin değiştirilmesi gerekebilir. Düzenlemeler bitirildiğinde tasarım yüzeyi Şekil 2.21'deki gibi görünmelidir (Freeman 2010).




**Şekil 2.20.** Label kontrolleri düzenlenmiş tasarım yüzü (Freeman 2010)

### 2.3.6. TextBox'ları ve NumericUpDown'ları düzenleme

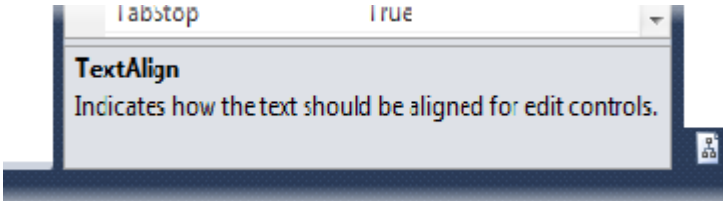
Pencerenin solunda başlangıç değeri olarak "0" gösterecek iki TextBox ve bunların textlerinin ekranın sağ tarafında hizalanması istenir. Tasarım yüzündeki birden fazla kontrole aynı değişikliğin uygulanması istendiğinde, onlar seçilir ve özellikleri düzenlenir. Değişiklikler seçilen kontrollerin tümüne etki edecektir. Çizelge 2.9'da gösterilen değişiklikler uygulanır (Freeman 2010).

**Çizelge 2.9.** Textbox kontrolleri için özellik değişimi (Freeman 2010)

| Property | Value |
|---|---|
| Text | 0 |
| TextAlign | Right |

Şekil 2.22'de TextBox kontrolünün TextAlign özelliği için gösterildiği gibi her bir özellik seçildiğinde Properties penceresinin alt kısmında özelliğin etkisinin ne olduğu

kısa bir mesajla gösterilir. Bu kullanılan özelliğin etkisinin ne olduğunu bulmanın en iyi yoludur.




**Şekil 2.21.** Kontrol özelliklerinin anlamını keşfetmek (Freeman 2010)

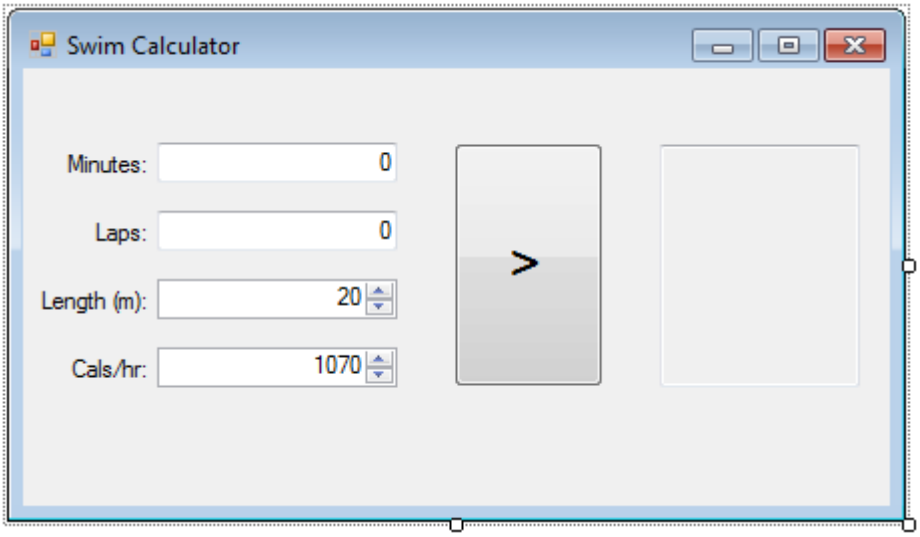
NumericUpDown kontrol TextBox gibidir fakat sadece sayısal değerler gösterilir ve kullanıcı kontrolün sağ tarafındaki kaydırma butonlarını kullanarak değeri değiştirebilir. Bu kontrol tipi, örnekte havuzun uzunluğu ve yüzmenin saatteki tükettiği kalori için kullanılacak. Bu değerler nadiren değiştirileceği için kullanılacak değerler program başlatıldığında varsayılan olarak görünecektir. Böylece her yüzmeye gidildiğinde bu değerleri değiştirmeye gerek kalmayacaktır. Çizelge 2.10 NumericUpDown kontrolüne yapılacak özellik değişimlerini listeler (Freeman 2010).

**Çizelge 2.10.** NumericUpDown kontrolleri için özellik değişimleri (Freeman 2010)

| Property | Value |
|---|---|
| TextAlign | Right |
| Minimum | 1 |
| Maximum | 2000 |
| Value | 20 (for the top control)<br>1070 (for the bottom control) |

İkinci NumericUpDown kontrolü için Value özelliğinden önce Maximum özelliğinin ayarlandığından emin olunmalıdır. Aksi taktirde 1070 değeri kontrolün varsayılan en üst limit değerini aşacaktır. Ekranın sağ tarafındaki büyük TextBox için ReadOnly özelliği "True" olarak değiştirilmelidir. Burası hesap sonuçlarının gösterileceği yer olup bu

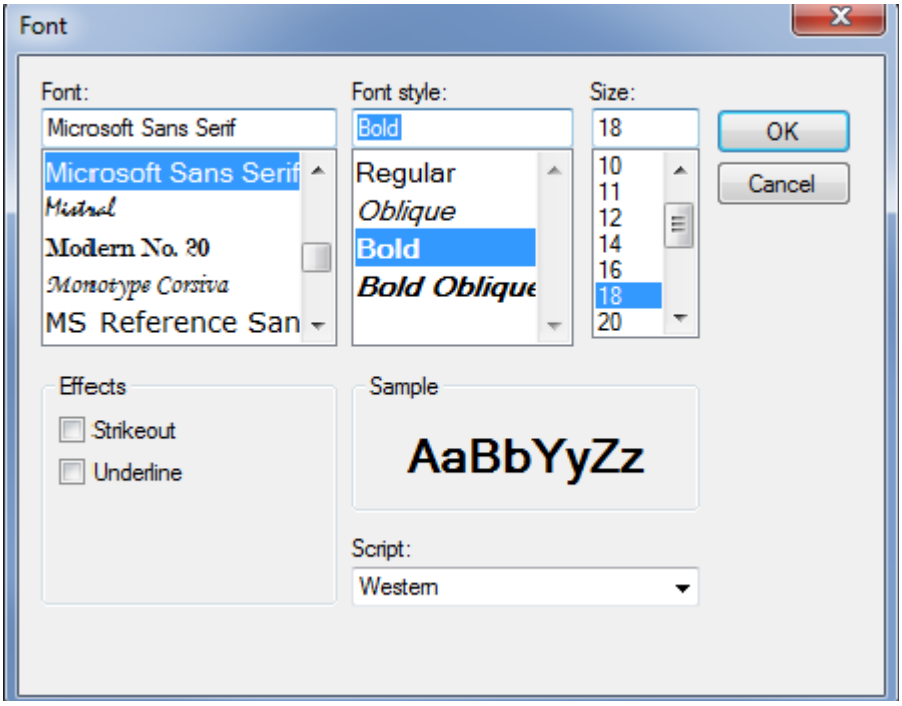
alanın kullanıcı tarafından değiştirilmesi istenmez. Bu kontroller değiştirildiğinde tasarım yüzü Şekil 2.23'tekine benzemelidir (Freeman 2010).




**Şekil 2.22.** TextBox ve NumericUpDown kontrolleri düzenlenmiş tasarım yüzü (Freeman 2010)

### 2.3.7. Düğme (Button) düzenleme

Buton kontrolüne iki değişiklik yapılması gerekir. Birincisi Text özelliğinin değerinin ">" karakteri olarak değiştirilmesidir. İkincisi yazı tipi boyutunun değiştirilmesidir. Eğer Font özelliği için değer tıklanırsa noktalı bir buton ortaya çıkacaktır. Bu buton tıklanırsa Şekil 2.24'te gösterildiği gibi Font seçme diyalog kutusu açılacaktır (Freeman 2010).




**Şekil 2.23.** Font diyalog kutusu (Freeman 2010)

Font diyalog kutusunun Size kısmından "18" ve Font Style kısmından "Bold" seçilir. Diyalog kutusunun kapanıp özellik değerinin değişmesi için "OK" butonu tıklanır.

### 2.3.8. MenuStrip ve StatusStrip ekleme

Birçok kontrol daha önceki kısımlarda ele alınan kontrollere benzer şekilde çalışır. Ancak, birbirinden çok az farklı olan iki kontrolün değiştirilmesine ihtiyaç duyulur. Birincisi program penceresinin üst kısmı boyunca standart menü sağlayan MenuStrip'tir.

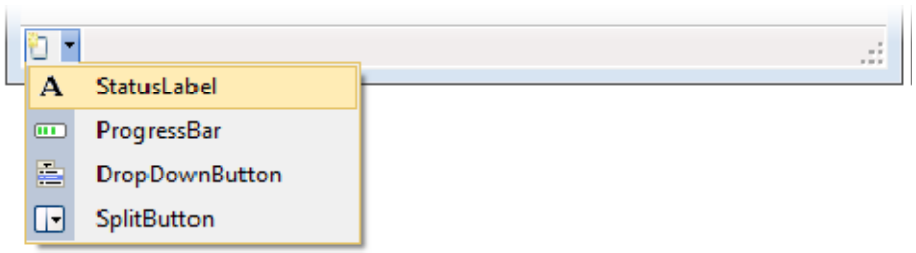
Başlamak için tasarım yüzündeki ana formun üst sol kısmına bir MenuStrip kontrolü sürüklenir (bu kontrolü bulmak için diğer kontrol gruplarının genişletilmesi gerekebilir). MenuStrip Şekil 2.25'te gösterildiği gibi pencereyi yatay olarak doldurmak için genişleyecektir.


**Şekil 2.24.** Ana pencereye bir MenuStrip ekleme (Freeman 2010)

Type Here kutusuna "File" girilir. Bu standart bir File menüsü oluşturur. Yazıldığı gibi eklenen kutular alt sağa açılacaktır. Bu yüzden o anki menüye yeni öğeler ya da yeni bir MenuStrip eklenebilir. File menüsünde sadece bir "Quit" öğesi istenildiği için Şekil 2.26'da gösterildiği gibi en aşağıdaki kutuya "Quit" girilir.

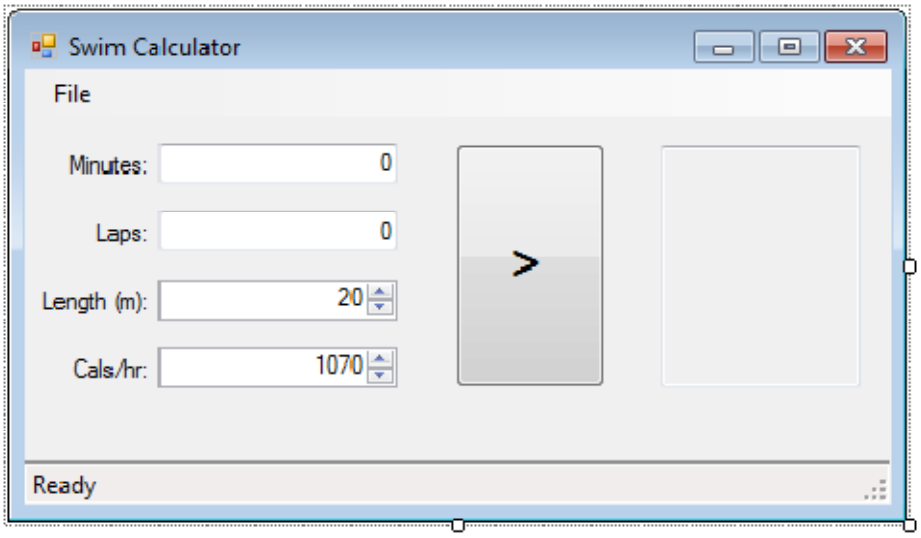

**Şekil 2.25.** Menüleri tanımla

StatusStrip kontrolü pencerenin altında durum çubuğu elde etmeye yarar. Ana pencerenin altına yerleşmesi için tasarım yüzüne bir StatusStrip kontrol sürüklenir. Şekil 2.27'da gösterildiği gibi kontroldeki aşağı açılan ok ikonu tıklanır ve açılan menüden StatusLabel'ı seçilir (Freeman 2010).




**Şekil 2.26.** Bir StatusStrip'te StatusLabel seçme (Freeman 2010)

StatusStrip farklı türde bilgi görüntüleyebilir. Sadece basit bir yazı etiketine ihtiyaç olduğu için StatusLabel seçeneği seçildi. Label eklendiğinde tıklanır ve Text özelliği "Ready" olarak değiştirilir. Bu öğeleri eklediğinde ve özellikleri değiştirdiğinde tasarım yüzü Şekil 2.28'dekine benzemelidir (Freeman 2010).



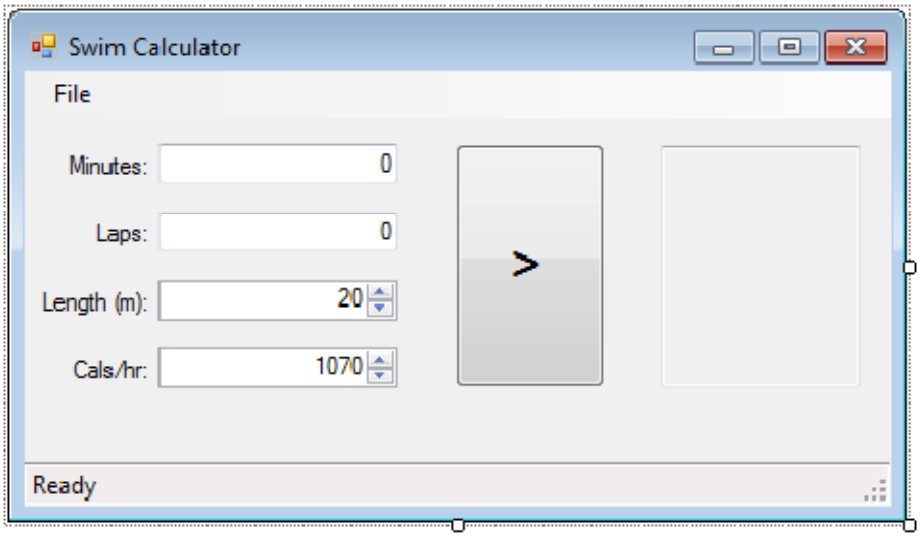

**Şekil 2.27.** Tüm kontroller eklendiğindeki tasarım yüzü (Freeman 2010)

### 2.3.9. Kontrol adlarını ayarlama

Tasarım yüzüne eklenen her bir kontrol için otomatik üye adı üretilir. Bazı kodlar yazıldığında kontrolleri adlandırmak gerekebilir ve otomatik üretilen adlar kontrolün amacından ziyade kontrol tipiyle adlandırılır. Örneğin TextBox kontrolleri textBox1, textBox2 v.b. şekilde adlandırılır. Bu bölümde daha sonra adlandırmak istediğimiz kontrollere anlamlı adlar atanacak. Örnek olarak Button kontrolü için kullanılan adı değiştirilsin. Tasarım yüzündeki button seçilir ve Design özellik grubundaki Name özelliği "convertButton" olarak değiştirilir. Form1.Designer.cs dosyasındaki tanımlanmış üyelerin kod ifaderine bakıldığında, adın aşağıdaki şekilde değiştiği görülür.

```
private System.Windows.Forms.Button convertButton;
```

Koddaki bu kontrolü adlandırmaya gelindiğinde, daha anlamlı adlar kullanılabilir. Şekil 2.29'de değiştirilmesi gereken kontrol isimleri ve her birine atanmış bir sayı gösterilir (Freeman 2010).




**Şekil 2.28.** Numaralandırışmış Kontroller (Freeman 2010)

Çizelge 2.11 Şekil 2.29'dan her bir kontrol için karşılık gelen numara ile birlikte kullanılması gereken ismi gösterir.

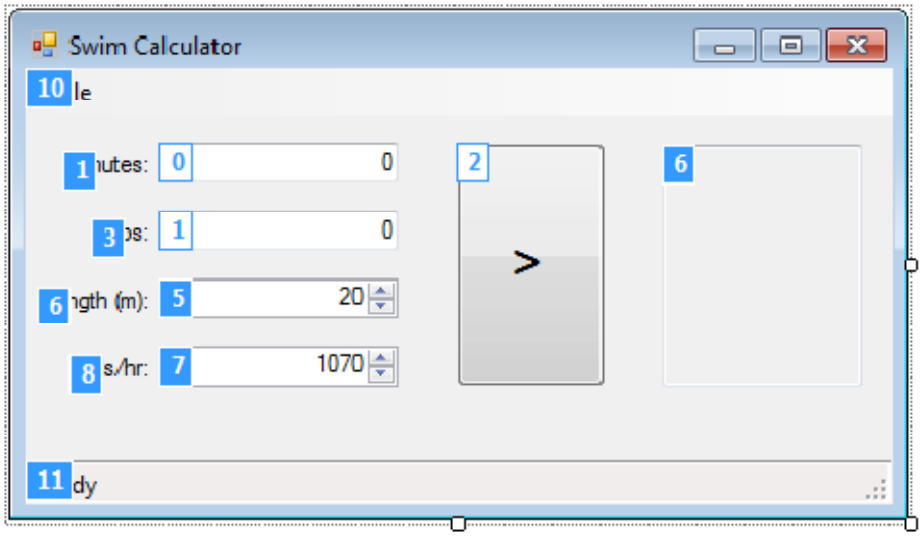
**Çizelge 2.11.** Kontrol Adları (Freeman 2010)

| Number | Type | Name |
|---|---|---|
| 1 | TextBox | resultsTextBox |
| 2 | Button | convertButton |
| 3 | TextBox | minutesTextBox |
| 4 | TextBox | lapsTextBox |
| 5 | NumericUpDown | poolLength |
| 6 | NumericUpDown | calsPerHour |
| 7 | ToolStripStatusLabel | statusLabel |

Kontrollerin ismini değiştirmeye gerek yoktur fakat kontrol adlarını kullanmaya gelindiğinde kodda hata yapma eğiliminin olduğu görülür. Hangi kontrolörün ne yaptığı unutulur. Kodda erişilmek istenen kontrolörün adını değişmek bir dakika alır fakat daha sonra saatlerce hata ayıklama eziyetinden kurtarabilir (Freeman 2010).

### 2.3.10. Sekme (Tab) sırasını ayarlama

Sıradaki adım, kontroller için sekme sırasının kullanıcı için anlamlı olmasından emin olmaktır. Sekme sırası tab tuşuna basıldığında hangi kontrolün seçileceğini belirler. Varsayılan sekme sırası kontrollerin tasarıma eklenme sırasıdır. Bir kontrol seçildiğinde basma gibi bir kullanıcı girişi olduğunda diğer kontrollerden herhangi birine yönlenir. Tab tuşu kontrolden kontrole seçimi taşımak için kullanılır. Örnekte en çok kullanılan üç kontrol; çalışma dakikasının girildiği TextBox, turların girildiği TextBox ve tıklandığında hesaplama yapacak olan buton arasında tab kullanılır. NumericUpDown kontrollerinde tab kullanılması istenmiyor. Çoğu zaman bunlar için varsayılan değerler düzeltilecektir. O anki sıralamayı görmek için View menüsünden Tab Order menü öğesi tıklanır. Tab tuşu kontrolleri seçerken kullandığı sıralamayı gösteren numaralanmış mavi kutular görülecektir. Sıralamayı değiştirmek için istenilen sırada mavi kutular tıklanır. Her bir kutu tıklandığında Şekil 2.30'da görüldüğü gibi rengi beyaza döner ve yeni sıra numarası gözükür (Freeman 2010).




**Şekil 2.29.** Sekme (Tab) sırasını değiştirme (Freeman 2010)

Şekil 2.30'dan görüldüğü gibi kontroller istenilen sırada tıklanıldı. Şekil 2.30'da gösterilen sırada 3 kontrol tıklanıldı ve daha sonra View menüsünden Tab View öğesi seçilerek tab sıralaması pasif edildi. Bir kontrolün TabStop özelliği "False" yapılarak tab tuşu ile seçilmesi durdurulabilir. Bu örnekte büyük TextBox ve NumericUpDown kontrolleri için bu yapıldı. Tab sıralamasını değiştirmeye gerek olmasa da şaşılacak sayıda kullanıcı formu tab tuşunu kullanarak yönlendirir. Kullanıcı bilgiyi almak isteyeceği bir tab sırasını kullanarak programı daha kolay yapabilir ve kullanım zorluklarını azaltabilir (Freeman 2010).

# 3. MATERYAL ve YÖNTEM

**Şekil 3.1.** Kullanılan donanımsal cihazlar

Tez'in bu kısmında tez hazırlanırken kullanılan materyal ve bu materyallerde kullanılan yöntemlerden bahsedilecektir. Tez ağırlıklı olarak programlama üzerine kurulmuş olsa da donanımsal olarak da kontrolör görevini icra etmek için bir PC, haberleşmek istenilen Anritsu firmasının MS9710B Optik Spektrum Analizörü ve PC ile cihaz arasında iletişimi sağlayacak olan Keithley firmasının KUSB-488B USB-GPIB dönüştürücüsü kullanılmıştır. PC ile optik spektrum analizörünün uzaktan kontrolü için, PC'de kullanılmak üzere bir arayüze ihtiyac vardır. Bu arayüz için Microsoft firmasının Visual C# programının 2010 Express Edition sürümü kullanılmıştır.

## 3.1. Arayüz Programı

Cihazın uzaktan kontrolünü sağlamak için kontrolör görevi gören PC'de bir arayüz programının olması gerekir. Bu arayüz programı için Microsoft firmasının Visual C# görsel programlama dili kullanılmıştır.

Görsel olarak bir arayüz oluşturulabilmesi için Visual C#'daki Form uygulaması kullanılmıştır. Cihazdan alınacak olan verilerin form üzerine grafiksel olarak yansıtılması için System.Drawing uzayında bulunan metot ve sınıflardan

yararlanılmıştır. System.Drawing uzayı (namespace) içinde çizim için kullanılan çok sayıda sınıf vardır.

Bunlardan bazıları şunlardır:

Graphics: Çizimi oluşturacak metotları içerir.
Pen: Çizim için kullanılacak kalemin rengini ve genişliği ayarlamak için kullanılır.
Brush: Fırçanın renk ve desenini ayarlamak için kullanılan sınıftır.
Color: Çizimde kullanılacak rengi ayarlamak için kullanılan sınıftır.
Font: Yazım için kullanılacak sınıftır.
Point: Çizimde kullanılacak koordinatları belirlemek için kullanılır.
Rectangle: Dikdörtgen alanı tanımlamak için kullanılır.
Path: Birden fazla çizimi tek bir çizim nesnesi altında toplamak için kullanılır.

Çizim için gerekli metotlar Graphics sınıfı içindedir. Diğer sınıflar çizim için gerekli ayarları yapmayı sağlar. System.Drawing isim uzayı projelere standart olarak eklenir. CreateGraphics() metodu ile grafik nesnesine ait yüzey oluşturulur. CreateGraphics() metoduyla oluşturulan nesne kullanıldıktan sonra Dispose() metoduyla bellekten silinmelidir. Grafiksel şekiller formun Paint() metodunda oluşturulur [Nesne tabanlı programlama].

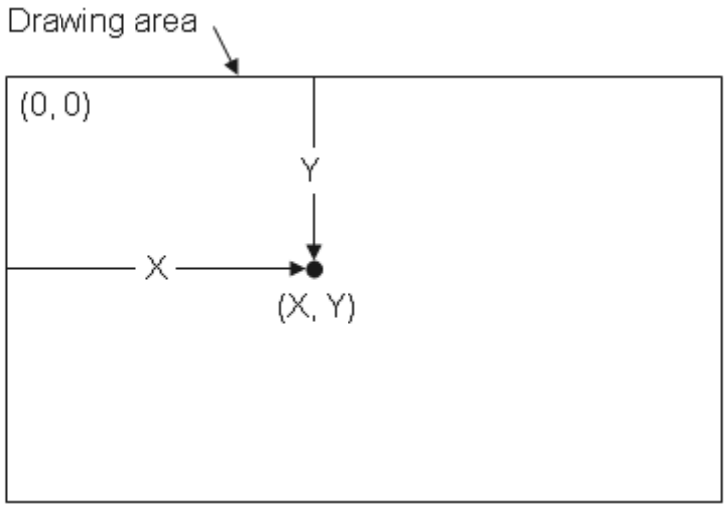
Visual C#, tüm bu araçları ihtiyaç olan herhangi çizim ve grafiği oluşturmak için sunar. Visual C# kullanıcıların yazı, çizgi, diktörgen, çember, elips, çokgen ve bir dizi diğer grafik şekillerini içeren değişken grafik objeleri çizmelerine izin veren bir GDI+(Graphic Device Interface – Grafik Cihaz Arayüzü) sınıf kütüphanesini kaynak olarak sunar. Burada Visual C#'da grafik çizmek için kullanılacak olan koordinat sistemi ile temel grafik şekilleri ve yüzeyi resim, renk gibi şeyleri doldurmada kullanılan Pen, Brush gibi çizim nesneleri ele alınacaktır (Xu 2007).

### 3.1.1. Koordinat sistemleri

Bir grafik nesnesi oluşturulmak istendiğinde grafik nesnesinin veya çizimin nerede gösterileceğinin tanımlanması gerekir. Bunu yapmak için Visual C#'ın grafik nesne koordinatlarını nasıl ölçtüğünün anlaşılması gerekir. Windows Form'u üzerindeki her bir nokta veya kontrol bir X ve Y koordinatına sahiptir. Aşağıdaki bölümde değişken koordinat sistemleri ve bunlar arasındaki ilişki ele alınacaktır (Xu 2007).

### 3.1.2. Varsayılan koordinat sistemi

Visual C# ve GDI+ grafik kütüphanesi iki boyutlu uzayda varsayılan üç koordinat sistemine sahiptir. Bunlar world (dünya), page (sayfa) ve device (cihaz)'dır. World koordinatları yeryüzündeki belirli bir grafiği modellemek için kullanılan koordinatlardır ve C# metotlarında geçen koordinatlardır. Page koordinatları bir form veya kontrol gibi çizim arayüzü tarafından kullanılan koordinat sistemi olarak adlandırılır. Device koordinatları bir ekran veya sayfa yaprağı üzerinde çizilecek gibi olan fiziksel cihazlar tarafından kullanılan koordinatlardır. C#'da bir noktadan (x1,y1) diğer bir noktaya (x2,y2) bir çizgi çizmek istenildiğinde, bu noktalar world koordinat sistemindedir. World koordinat sisteminde mesafeyi ölçmede kullanılacak birim uygulamaya göre tanımlanabilir. Grafik nesneleri world koordinat sisteminde doğrudan bilgisayar ekranına çizilemez. Ekranda bir grafik nesnesi çizmeden önce, koordinatlar sıralı bir şekilde dönüşüme uğramalıdır. Bu dönüşüm world dönüşüm olarak adlandırılır ve world koordinatlarını page koordinatlarına dönüştürür; diğer dönüşüm page dönüşüm olarak adlandırılır ve page koordinatlarını device koordinatlarına dönüştürür. Üç koordinat sisteminde de varsayılan orjin noktası (0,0) olup çizim alanının üst sol köşesinde konumlandırılmıştır. X koordinatı çizim alanının sol kenarından noktaya kadar olan uzaklıkla ve Y koordinatı çizim alanının üst kısmından noktaya kadar olan uzaklıkla ifade edilir. Şekil 3.2. bir noktanın X ve Y koordinatlarının çizim alanı ile ilişkisinin nasıl olduğunu gösterir (Xu 2007).

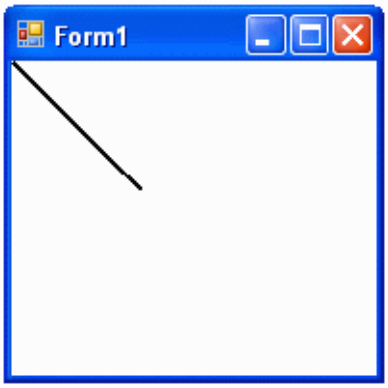



**Şekil 3.2.** Varsayılan C# koordinat sistemi (Xu 2007)

Üç koordinat sistemi içinde varsayılan birim pikseldir. Koordinat sistemi, orjin başka bir yere kaydırılarak ve farklı bir ölçüm birimi ayarlanarak isteğe göre özelleştirilebilir. Örneğe bakılarak bunun nasıl yapıldığı görülür. Öncelikle Microsoft Visual Studio açılıp File>New>Project'ten yeni bir Windows Forms Application projesi oluşturuldu. Proje Example1_1 olarak adlandırıldı. Form1'in OnPaint metotunu geçersiz kılarak elde edilebilen redraw fonksiyonuna sahip olması istenir. İlk başta (0,0) noktasından (1,1) noktasına inç birimiyle bir çizgi çizildi. Bunu yapan kod listesi aşağıdaki gibidir (Xu 2007).

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace Example1_1
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
```

```
                this.SetStyle(ControlStyles.ResizeRedraw, true);
                this.BackColor = Color.White;
            }
            protected override void OnPaint(PaintEventArgs e)
            {
                Graphics g = e.Graphics;
                //Aşağıdaki kodlar inç biriminde (0,0)'dan (1,1)'e bir çizgi çizer.
                g.PageUnit = GraphicsUnit.Inch;
                Pen blackPen = new Pen(Color.Black, 1 / g.DpiX);
                g.DrawLine(blackPen, 0, 0, 1, 1);
            }
        }
}
```



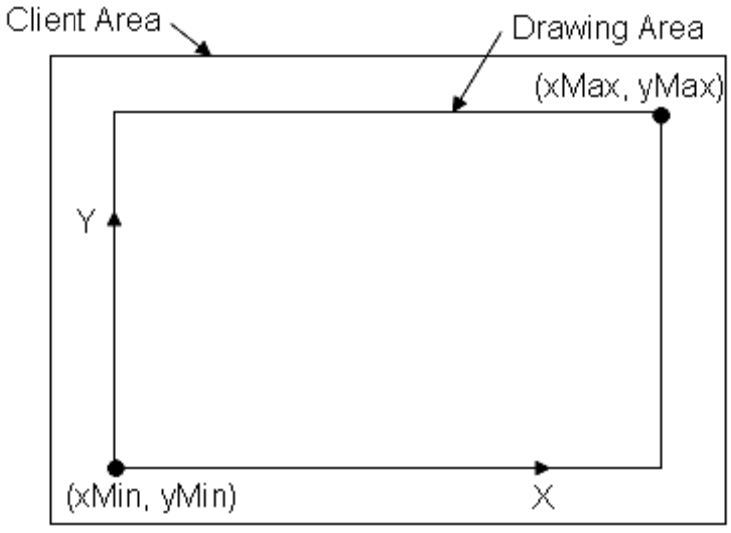

**Şekil 3.3.** (0,0) noktasından (1,1) noktasına bir çizgi çizme (Xu 2007)

Şekil 3.3'te sonuç grafiği gösterilmektedir. Form1 sınıfının kod listesinden PageUnit özelliğinin ölçüm biriminin bir inç olduğunu belirtmek için GraphicsUnit.Inc olarak ayarlandığı görülebilir. Daha sonra bir kalem nesnesi oluşturuldu ve kalemin genişliği 1/g.DpiX ayarlandı. Grafik sınıfının DpiX özelliği bu grafik nesnesi tarafından desteklenen yatay çözünürlük için her bir inçteki nokta sayısını gösteren bir değerdir. Bu gereklidir. Çünkü mecvut PageUnit ayarları, ayarlanan Grafik nesne şekline bağlı olarak bir birimlik kalemin bir piksel, bir milimetre, bir nokta veya bir inç çizgi

çizmesine etki edecektir. Eğer kalem genişliği bu şekilde ayarlanmazsa, bir inç kalınlığında bir çizgi çizilmiş olur. Form1 tekrar boyutlandırıldığında grafiğin tekrar çizildiğinden emin olmak için Form1 yapıcısı içindeki ControlStyles.ResizeRedraw'ın "true" olarak ayarlandığına dikkat edilmelidir (Xu 2007).

### 3.1.3. Koordinatları özelleştirme

Bir uygulama standart koordinat sistemine ek olarak kendi koordinat sistemini tanımlayabilir. Burada iki boyutlu şekil ve grafik uygulamaları için bir özel koordinat sistemi kullanılmıştır. Bu koordinat sistemi gerçek dünyadaki grafik nesnelerinin birimlerinden bağımsızdır ve Y ekseni birçok grafik uygulamasında olduğu gibi alttan üste doğrudur. Bu sistemde page koordinat sistemi cihaz sistemi için aynıdır ve her iki sistem de pikseldir. Bu özelleştirilmiş sistem Şekil 3.4'te gösterilmiştir (Xu 2007).




**Şekil 3.4.** Özelleştirilmiş koordinat sistemi (Xu 2007)

Ana alan(Client Area) içerisinde kaydırılmış bir kenarla çizim alanı(Drawing Area) tanımlanmıştır. Geleneksel X-Y koordinat sistemi çizim alanında tanımlanabilir. C# ile bunun nasıl elde edileceği aşağıdaki örnekle gösterilmiştir.

Yeni bir Windows Application projesi oluşturulur ve Example1-2 olarak adlandırılır. Form1.cs'nin kod listesi aşağıdaki gibidir.

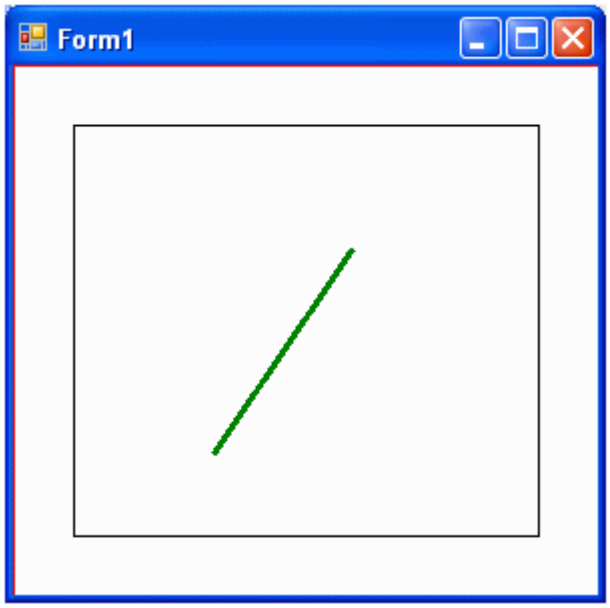
```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace Example1_2
{
        public partial class Form1 : Form
        {
                // Çizim alanını tanımla.
                private Rectangle PlotArea;
                // World koordinat sisteminde birimi tanımla.
                private float xMin = 0f;
                private float xMax = 10f;
                private float yMin = 0f;
                private float yMax = 10f;
                // Pikseldeki kaymayı (offset)'i tanımla.
                private int offset = 30;
                public Form1()
                {
                        InitializeComponent();
                        this.SetStyle(ControlStyles.ResizeRedraw, true);
                        this.BackColor = Color.White;
                }
                protected override void OnPaint(PaintEventArgs e)
                {
                        Graphics g = e.Graphics;
                        // Çizmek istenilen grafikteki çizim alanını ve konumu hesapla
                        Rectangle rect = ClientRectangle;
                        PlotArea = new Rectangle(rect.Location, rect.Size);
```

```
                PlotArea.Inflate(-offset, -offset);
                //Kalem kullanarak ClientRectangle ve PlotArea çiz:
                g.DrawRectangle(Pens.Red, rect);
                g.DrawRectangle(Pens.Black, PlotArea);
                // (3,2) noktasından (6, 7) noktasına
                // 3 piksel genişliğindeki bir kalemle bir çizgi çiz.
                Pen aPen = new Pen(Color.Green, 3);
                g.DrawLine(aPen, Point2D(new PointF(3, 2)), Point2D(new
                PointF(6, 7)));
                aPen.Dispose();
                g.Dispose();
            }
            private PointF Point2D(PointF ptf)
            {
                PointF aPoint = new PointF();
                aPoint.X = PlotArea.X + (ptf.X - xMin) *
                PlotArea.Width / (xMax - xMin);
                aPoint.Y = PlotArea.Bottom - (ptf.Y - yMin) *
                PlotArea.Height / (yMax - yMin);
                return aPoint;
            }
        }
}
```

Bu örneğe, çizim alanının boyutlarını gösteren bir PlotArea üyesinin yanısıra özel koordinat eksenlerinin minumum ve maksimum değerlerini tutmak için üye alanların oluşturulması ile başlanıldı. xMin, xMax, yMin, ve yMax değerleri uygulamanın isteğine bağlı olarak değiştirilerek herhangi bir boyutta çizim alanı oluşturulabilir. xMin, xMax, yMin, ve yMax'ın world koordinat sisteminde tanımlanmış gerçek dünya birimlerinde olması gerektiğinden emin olunmalıdır.

Daha sonra başlangıçta Form1'in ana alanıyla(client area) aynı boyutta çizim alanı tanımlandı. Grafiğin Form1'in tüm alanını doldurması istenmediğinden daha sonraki uygulamalarda X ve Y ekseni için etiketler eklendiğinde daha etkili bir şekilde kaydırma yapabilmek için "Inflate" metodu kullanılacaktır. Daha sonra DrawRectangle metodu kullanılarak ana ve çizim dikdörtgenlerinin ana hatları belirtildi.

World koordinat sisteminin birimlerinden bağımsız olması gereken bir çizim alanında grafik nesnelerinin nasıl çizileceği konusuna dikkat edilmesi gerekir. Burada world koordinat sistemindeki noktaları device koordinat sistemine dönüştürmek için Point2D metodu kullanıldı. Bu dönüşümden sonra Pen(kalem) ve Brush(fırça) dahil tüm grafik nesneleri için birim pikseldir. World koordinat sistemindeki herhangi birimi otomatik device koordinat sistemine dönüştürüp geriye daima piksel biriminde getiren Point2D metodu basitçe anlatılmıştır. Bu işlem asla page koordinat sistemine dokunmaz. Eğer page koordinat sistemindeki PageScale ve PageUnit gibi özellikler değiştirilirse, beklenmedik sonuçlar alınır. Bu nedenle özelleştirilmiş koordinatlar kullanılırken page koordinat sistemindeki hiçbir şey değiştirilmemelidir (Xu 2007).




**Şekil 3.5.** Özel koordinat sisteminde (3, 2) noktasından (6, 7) noktasına bir çizgi çizme (Xu 2007).

Point2D metodu şu şekilde çalışır. Başlangıçta world koordinat sistemindeki bir noktanın tüm X bileşenleri aşağıdaki formül kullanılarak dönüştürülür.

```
aPoint.X = PlotArea.X + (ptf.X - xMin) * PlotArea.Width / (xMax - xMin);
```

Cihaz koordinat sisteminde, doğru pozisyondaki özel koordinatların orjininin X bileşenini yerleştirmek için gerekli bir kaydırma olan piksel birimiyle PlotArea.X aPoint'in X bileşenine eklendi. Daha sonra ölçeklendirme gerçekleştirildi. World koordinat sisteminde aynı birime sahip (ptf.X – xMin) ve (xMax – xMin)'in her ikisi de bölünerek dengelendi. Bu yüzden bu ölçeklendirme ifadesinin birimi yalnızca piksel olan PlotArea.Width ifadesi tarafından tanımlandı. Yukarıdaki dönüşümün sadece doğru birim elde etmeyip aynı zamanda cihaz koordinat sisteminde doğru pozisyon elde ettiği kolaylıkla kontrol edilebilir (Xu 2007).

Y bileşeni için dönüşüm biraz farklıdır. Sadece ölçeklendirme işleminin gerçekleştirilmesi yetmiyor aynı zamanda cihaz koordinat sistemindeki Y ekseninin ters çevrilmesi gerekiyor. Aşağıfaki formüş Y bileşeni dönüşümü için kullanılır.

```
aPoint.Y = PlotArea.Bottom - (ptf.Y - yMin) * PlotArea.Height / (yMax - yMin);
```

Şekil 3.5'te gösterildiği gibi çizim alanında (3,2) noktasından (6,7) noktasına bir çizgi çizildi. Bu çizginin son noktası world koordinat sisteminde tanımlanan birimdedir. Bu noktalar çizgiyi çizmede doğrudan kullanılmadı, bunların yerine Point2D metodu sayesinde dönüştürülen noktalar kullanıldı. Bu çizgi 3 piksel genişliğinde yeşil bir kalemle (Pen) çizildi.

```
Pen aPen = new Pen(Color.Green, 3);
g.DrawLine(aPen, Point2D(new PointF(3, 2)), Point2D(new PointF(6, 7)));
```

Kalem genişliğininin biriminin daime piksel olduğu açıkça görülür. World koordinat sisteminde kullanılan birime bakılmaksızın Pen ve Brush için herhangi bir dönüşüm gerçekleştirmeye gerek yoktur (Xu 2007).

### 3.1.4. Pencere (Window) ve Ekran Penceresi (Viewport)

Bir grafik nesnesi sınırları belli bazı soyut yer olan kendi koordinat sisteminde tanımlanmış kabul edilir. Örneğin, X veri aralığı 0-10 ve Y değerleri 50-100 arasında olan basit bir X-Y grafiği oluşturulmak istensin. 0<=X<=10 ve 50<=Y<=100 aralığında bir koordinat sisteminde çalışılabilir. Pratikte genelde bir grafiğin tamamıyla değil bir kısmı ile ilgilenilir. Bu yüzden world koordinat sisteminde ilgilenilen belirli alan tanımlanabilir. Bu ilgili alan "Window" olarak adlandırılır. Grafik nesnelerini ekranda çizmek için bu "Window"u koordinat sistemine eşlemek gerekir. Cihaz koordinat sistemindeki bu eşlenmiş "Window" bir "Viewport" olarak adlandırılır (Xu 2007). Önceki bölümde özel koordinat sisteminde X ve Y eksenleri için sınırlar tanımlanmıştı. Örneğin;

```
private float xMin = 0f;
private float xMax = 10f;
private float yMin = 0f;
private float yMax = 10f;
```

Bu özel koordinat sisteminde ilgili kısım tanımlanır ve bu ilgili alan "Window" olarak çağrılır. Gösterilmek istenenin ne olduğu bilindiğinde, onun bilgisayar ekranında nerede gösterileceğine karar vermek gerekir. Example1_2'de cihaz koordinat sisteminde grafik nesnelerini göstermede kullanılan bir ekran alanı oluşturmak için PlotArea tanımlandı. Bu PlotArea ViewPort olarak adlandırıldı.

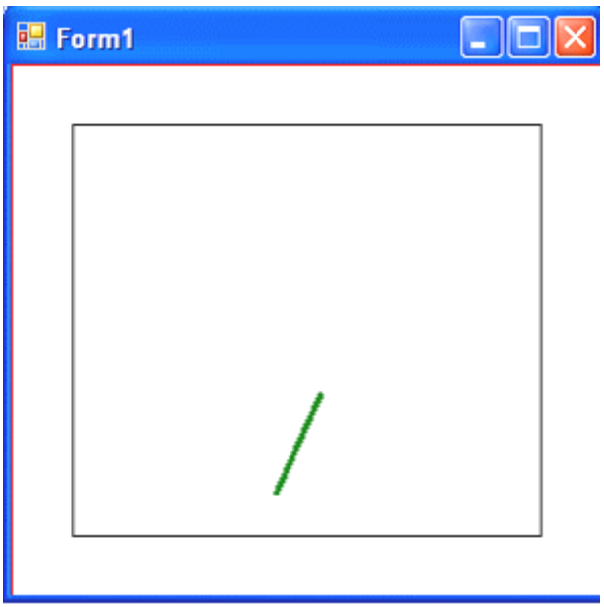
Bu Viewport ekrandaki grafik nesneleri kolayca görülebilir, konum ve boyutu ayarlamakta kullanılabilir. Viewport'u değiştirmek ekrandaki grafik nesnelerinin

görünmesine etki eder. Bu efektler “Zooming” ve “Panning” olarak adlandırılır (Xu 2007).

### 3.1.5. Zumlama (Zooming)

“Window”un boyut ve pozisyonu grafik nesnesinin hangi parçasının çizileceğini belirler. Window ve viewport’un ilgili büyüklüğü ekranda gösterilecek grafik nesnelerindeki ölçeği belirtir. Orantılı olarak büyük bir pencere küçük grafik nesneleri üretirken, aynı mantıkla küçük bir pencere büyük grafik nesneleri üretir. Bu yüzden “zoom out (uzaklaşma)” efekti görülmesi için pencerenin boyutu xMin, xMax, yMin ve yMax parametreleri değiştirilerek artırılır. Example1_2’de ele alınan değerler değiştirilerek uygulama çalıştırılırsa Şekil 3.6’daki sonuç elde edilir (Xu 2007).

```
private float xMin = -10f;
private float xMax = 20f;
private float yMin = 0f;
private float yMax = 20f;
```

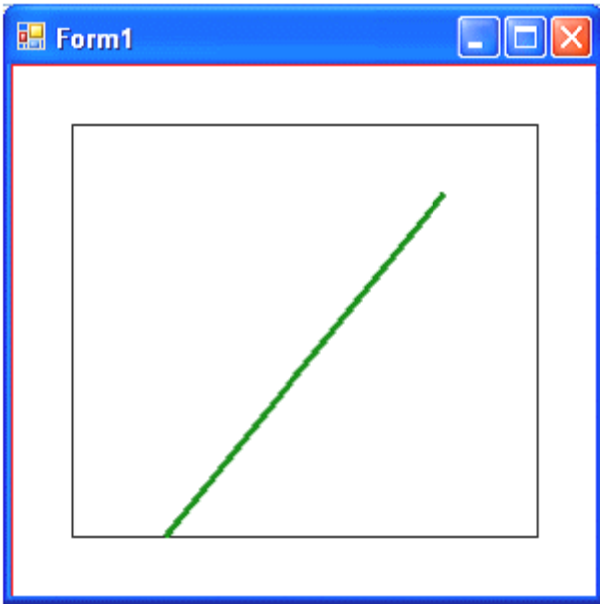


**Şekil 3.6.** Pencerenin boyutu artırılarak çizginin konum ve boyutu değiştirildi (Zoom out - Uzaklaşma) (Xu 2007).

Diğer taraftan eğer pencerenin boyutu küçültülürse, çizgi ekranda daha büyük görünür. Aşağıdaki gibi Example1_2 parametreleri değiştirilerek bir "zoom in (yaklaşma)" efekti elde edilebilir.

```
private float xMin = 2f;
private float xMax = 7f;
private float yMin = 2f;
private float yMax = 8f;
```

Program çalıştırılırsa Şekil 3.7'deki sonuç elde edilir.



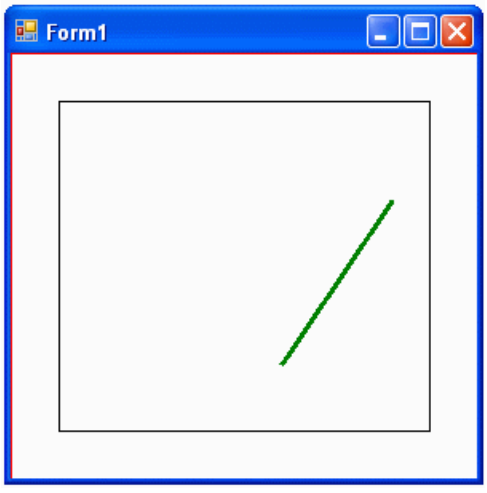

**Şekil 3.7.** Pencere boyutu küçültülerek çizginin konum ve boyutu değiştirildi (Zoom in – Yakınlaştırma) (Xu 2007).

### 3.1.6. Kaydırma (Panning)

Panning pencereyi kaydırarak ekrandaki tüm grafik objelerinin taşınması olarak tanımlanır. Bir panning işleminde pencere boyutu değişmeden kalır. Örneğin, Example1_2'deki kodlar değiştirilerek pencere sola taşınabilir (Xu 2007).

```
// Orijinal pencereyi 3 birim sola taşıma
private float xMin = -3f ;
private float xMax = 7;
private float yMin = 0f;
private float yMax = 10f;
```

Çizgiyi çizim alanının sağ tarafına doğru taşımayla eşit olan bu işlem Şekil 3.8'deki gibi bir sonuç üretir.

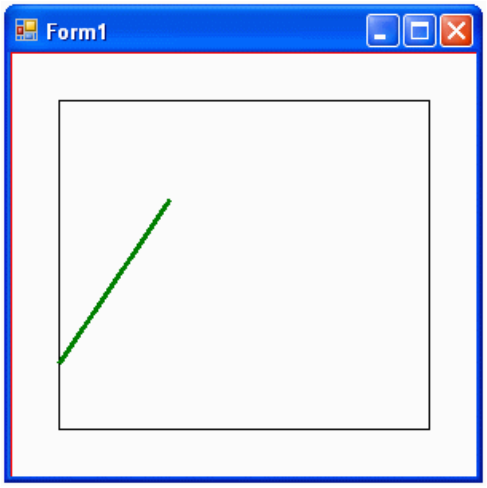



**Şekil 3.8.** Pencereyi sola doğru taşıma (Xu 2007)

Diğer taraftan, eğer aşağıdaki kodlar kullanılarak pencere sağa taşınırsa Şekil 3.9'daki sonuç elde edilir:

```
// Orijinal pencere 3 birim sağa taşınırsa
```

```
private float xMin = 3f ;
private float xMax = 13;
private float yMin = 0f;
private float yMax = 10f;
```

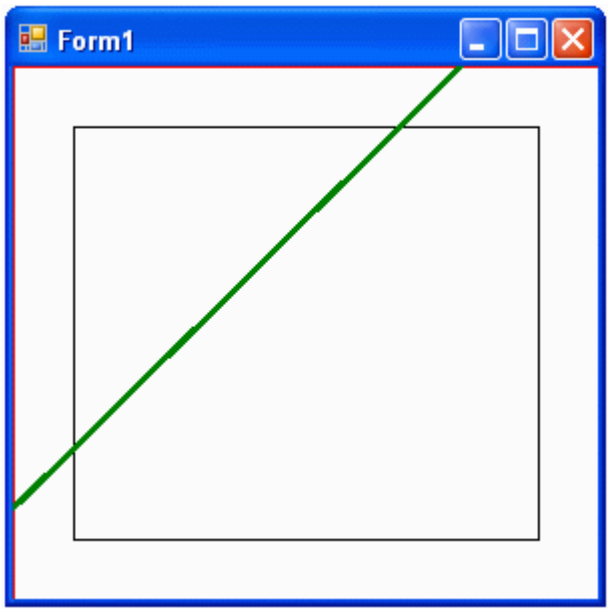



**Şekil 3.9.** Pencereyi sağa doğru taşıma (Xu 2007)

Example1_2'deki kod kullanıldığında "yakınlaştırma-zooming in" konusunda sonuca dikkat edilmelidir. Çok fazla yakınlaştırma yapıldığında istenmeyen bir durum meydana gelebilir. Örneğin, aşağıdaki kodla yakınlaştırma yapmak istenilirse:

```
private float xMin = 4f;
private float xMax = 6;
private float yMin = 3f;
private float yMax = 6f;
```

Şekil 3.10'da gösterildiği beklenmedik bir sonuç meydana geldi. Yani, çizgi çizim alanının dışında çizildi (Xu 2007).




**Şekil 3.10.** Çizgi yakınlaştırma yapılarak çizim alanının dışına çizildi (Xu 2007).

Bu sorundan sakınmak amacıyla geçerli çizim alanına yerleştirmek için bir kullanıcı kontrolü kullanılabilir. Bunu yapmak için C# Windows Application projesi oluşturuldu ve projeye Example1_3 adı verildi. Toolbox'tan bir Panel kontrolü seçilip Form1 üzerine sürüklendi ve drawingPanel olarak tekrar adlandırıldı. Form1.cs'nin kod listesi aşağıdaki gibidir (Xu 2007):

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace Example1_3
{
        public partial class Form1 : Form
        {
                // Unit defined in world coordinate system:
                private float xMin = 0f;
                private float xMax = 10f;
                private float yMin = 0f;
```

```
private float yMax = 10f;
// Unit in pixel:
private int offset = 30;
public Form1()
{
        InitializeComponent();
        this.SetStyle(ControlStyles.ResizeRedraw, true);
         this.BackColor = Color.White;
        // Subscribing to a paint eventhandler to drawingPanel:
        drawingPanel.Paint+=newPaintEventHandler(drawingPanelPaint)
        ;
        drawingPanel.BorderStyle = BorderStyle.FixedSingle;
        drawingPanel.Anchor = AnchorStyles.Bottom;
        drawingPanel.Anchor = AnchorStyles.Left;
        drawingPanel.Anchor = AnchorStyles.Right;
        drawingPanel.Anchor = AnchorStyles.Top;
}
private void drawingPanelPaint(object sender, PaintEventArgs e)
{
        drawingPanel.Left = offset;
        drawingPanel.Top = offset;
        drawingPanel.Width = ClientRectangle.Width - 2 * offset;
        drawingPanel.Height = ClientRectangle.Height - 2 * offset;
        Graphics g = e.Graphics;
        Pen aPen = new Pen(Color.Green, 3);
        g.DrawLine(aPen, Point2D(new PointF(2, 3)), Point2D(new
        PointF(6, 7)));
        aPen.Dispose();
        g.Dispose();
}
private PointF Point2D(PointF ptf)
```

```
        {
            PointF aPoint = new PointF();
            aPoint.X = (ptf.X - xMin) * drawingPanel.Width / (xMax -
            xMin);
            aPoint.Y = drawingPanel.Height - (ptf.Y - yMin)*
            drawingPanel.Height / (yMax - yMin);
            return aPoint;
        }
    }
}
```

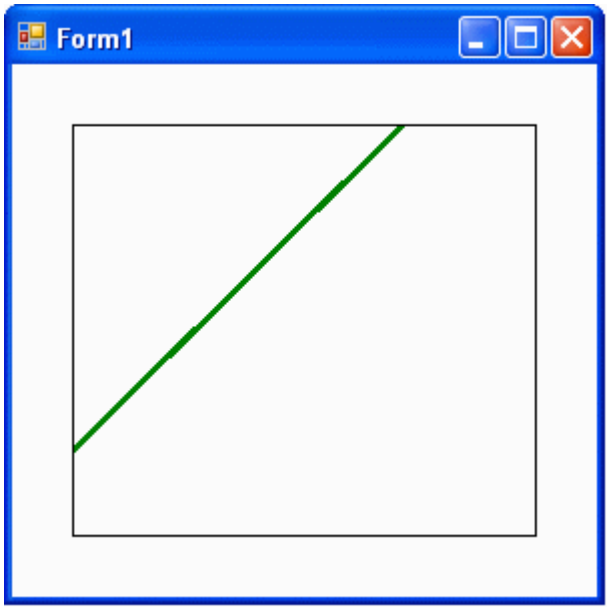
Bu kontrol panelinin Paint event handler'inin içine birşeyler çizebilmek için, Paint olayının çağrıldığı her anın haber verilmesinin gerektiğini Windows'a söylemek gerekir. Olaya yazarak ya da paint olayı geçersiz kılınarak bir kontrolden miras alınması (inherit) durumunda bu yapılabilir. Aşağıdaki sözdizimi Form1 yapıcısının içinde kullanılarak Paint event (olay) yazıldı (Xu 2007).

```
drawingPanel.Paint += new PaintEventHandler(drawingPanelPaint);
```

Bu şekilde aynı adla tanımlanmış bir drawingPanelPaint yöntemi uygulandı. Burada yapılması istenilen diğer bir husus Example1_2'den biraz farklı Point2D ölçeklendirmesi. Bunun sebebi Panel kontrolünün orjininin her zaman bu kontrolün üst sol köşesine yerleştirilmesidir. Eğer program çalıştırılırsa beklendiği gibi Şekil 3.10'da gösterilen sonucun aynısı meydana gelecektir. Eğer aşağıdaki kod listesindeki parametreler değiştirilirse (Xu 2007):

```
private float xMin = 4f;
private float xMax = 6;
private float yMin = 3f;
private float yMax = 6f;
```

Şekil 3.11'deki sonuç elde edilir.




**Şekil 3.11.** Çizgi çok fazla yakınlaştırma olsa da daima drawingPanel içine çizilir (Xu 2007).

Burada Şekil 3.10. ve Şekil 3.11. arasındaki fark açıkça görülebilir.

### 3.1.7. Kalem (Pen) ve Fırça (Brush)

Grafik nesneleri uygulama programı ve bilgisayar ekranı arasında bir arayüz sağlar. Bir grafik nesnesi oluşturulduktan sonra çizgileri çizmek ve şekilleri doldurmak için kullanılabilir. Ancak, Grafik sınıfı kullanmak için bir düzlem sağlamasına rağmen, çizim yapabilmek için bir araca ihtiyaç vardır. Pen (kalem) ve Brush (fırça) sıklıkla kullanılan iki araçtır. GDI+ kütüphanesi her biri ayrı Pen ve Brush sınıflarından Pen ve Brush araçlarını içerir. Pen sınıfı düz çizgi, eğri çizgi ve şekillerin dış çizgilerini çizmek için kullanılır. Brush sınıfı şekilleri renk ve şablonlarla doldurmak için kullanılır. Bir önceki bölüme bakılırsa Pen sınıfının koordinat sistemini göstermek için kullanıldığı görülebilir (Xu 2007).

### 3.1.7.a. Kalem (Pen) sınıfı

Pen sınıfı özel renk ve kalınlığa sahip özel kalemler oluşturmada kullanılır. Bir pen nesnesi tarafından çizilen çizgi katı renk ve dokuları içeren çeşitli dolgularla doldurulabilir. Dolgu biçimi dolgu nesnesi olarak kullanılan fırça veya dokuya bağlıdır. Pen sınıfı için dört farklı tip yapıcı (constructor) vardır: iki yapıcı fırça nesnesi yerine renk tanımlamaya izin verir ve iki yapıcı kalem kalınlığı tanımlama seçeneği sunar. Bu dört yapıcı aşağıdaki gibidir (Xu 2007):

Belirli bir renkte Pen sınıfının bir örneği (instance) oluşturma:
public Pen(color);
Belirli bir fırça ile Pen sınıfının bir örneğini oluşturma:
public Pen(brush);
Belirli bir fırça ve kalınlıkta Pen sınıfının bir örneğini oluşturma:
public Pen(brush, float);
Belirli bir renk ve kalınlıkta Pen sınıfının bir örneğini oluşturma:
public Pen(color, float);

Bu yapıcılarda color bir renk nesnesini, float kalınlık için float tipinde bir değeri ve brush bir fırça nesnesini temsil eder. Pen sınıfı kalem nesnesinin grafik öğelerini nasıl çizeceğinin büyük miktarda kontrolünü sunan birkaç özelliğe sahiptir. En sık kullanılan özellikler şunlardır (Xu 2007):

• Alignment (hizalama) – Kalem nesnesi için hizalama alır veya ayarlar.
• Brush (fırça) – Kalem nesnesinin özelliklerini belirleyen fırça nesnesini alır veya ayarlar.
• Color (renk) – Kalem nesnesinin rengini alır veya ayarlar.
• DashStyle (çizgi biçimi) – Kalem nesnesi ile çizilen kesikli çizgi için çizgi stilini alır veya ayarlar.
• Width – Kalem nesnesinin kalınlığını alır veya ayarlar.

### 3.1.7.b. Fırça (Brush)sınıfı

Fırçalar şekilleri renk, şablon ve resimlerle doldurmak için kullanılır. Fırça sınıfı bir özel temel sınıftır ve örneklendirilemez. Bir brush (fırça) nesnesi oluşturmak için onun üretilen sınıflarını kullanmak gerekir ki onlar şunlardır (Xu 2007):

• SolidBrush – Bu sınıf tek bir renkten oluşan fırçayı tanımlar.
• TextureBrush – Bu sınıf bir şeklin içini resim doldurmak için kullanılan fırçayı tanımlar.
• HatchBrush – Bu sınıf tarama biçimi, ön plan rengi ve arka plan rengiyle bir diktörtgen fırça tanımlar.
• LinearGradientBrush – Bu sınıf iki renk gradient ve özel çoklu renk gradientleri içerir.
• PathGradientBrush – Bu sınıf bir gradientli GraphicsPath nesnesin içini dolduran fırça nesnesini içerir (Xu 2007).

```
// Kırmızı renkli bir SolidBrush oluşturma:
SolidBrush sb = new SolidBrush(Color.Red);
// Bir resim dosyası (myImage.gif) ile TextureBrush oluşturma:
Bitmap bmp = new Bitmap(“myImage.gif”);
TextureBrush tb = new TextureBrush(bmp);
// Bir Cross şablonlu HatchBrush oluşturma:
HatchBrush hb = new HatchBrush(HatchStyle.Cross,Color.Black, Color.White);
//İki noktadan tanımlanmış bir Gradientli LinearGradientBrush oluşturma:
LinearGradientBrush lgb = new LinearGradientBrush(New Point(10, 10),
new Point(100, 30), Color.Red, Color.Black);
//Bir Grafik yolu için PathGradientBrush oluşturma:
PathGradientBrush pgb = new PathGradientBrush(graphicsPath);
pgb.CenterColor = Color.Red;
pgb.SurroundColors = new Color[] {Color.Black};
```

Yukarıdaki kodlar grafik şekilleri doldurmaya hazır çeşitli fırçalar oluşturur. PathGradientBrush için yapıcı bir GraphicsPath geçildi. Daha sonra pgb.CenterColor özelliği kırmızı renk olarak ayarlandı. Daha sonraki ifadeye dikkat edilirse;

```
pgb.SurroundColors = new Color[] {Color.Black};
```

Burada SurroundColors özelliği için bir dizi renk geçmektedir. Bu durumda sadece bir renkli dizi oluşturulur, böylece bu fırça grafik kenarını siyaha boyayarak merkezde kırmızıya geçer (Xu 2007).

### 3.1.8. Temel grafik şekilleri

C#'daki Windows Formları bir uygulamanın tipik kullanıcı arayüzündeki en büyük birimdir. Grafik şekilleri doğrudan Form üzerine çizilebilir. Form aynı zamanda Example1_1 ve Example1_2'de görüldüğü gibi diğer kontrolleri de içerebilir. Grafik nesnelerinin bir panel kontrolünün üzerine çizildiği Example1_3'te gösterildiği gibi kontrollere de form gibi doğrudan grafik nesneleri çizilebilir. Eğer çizim formun herhangi bir kısmına taşınmak istenirse, basitçe panel kontrolünün yerinin değiştirilmesi yeterlidir. Aksine Form üzerine doğrudan çizilmiş bir grafik çizimini taşımak için çizim kaynak kodlarının değiştirilmesi gerekir. C# ve GDI+ kütüphanesindeki bazı temel grafik şekilleri sonraki bölümlerde ele alınacaktır (Xu 2007).

### 3.1.9. Noktalar (Points)

GDI+ ve C#'da Point ve PointF olmak üzere iki nokta yapısı vardır. Point yapısı 2D düzlemde tanımlanmış X ve Y koordinatlarının sıralı tamsayı çiftlerini gösterir. Point yapıcısı (constructor) bir Point nesnesinin bir tamsayıdan, bir boyuttan veya iki tamsayıdan oluşturulmasına olanak sağlar (Xu 2007).

• Point() – Bir nokta nesnesi oluşturur ve X ve Y veri üyelerine sıfır değerini atar. Bu varsayılan yapıcıdır.
• Point(Size) – X ve Y veri üyelerine başlangıç değeri atamak için bir Size nesnesi kullanarak bir nokta nesnesi oluşturur.
• Point(int, int) – X ve Y veri üyelerine başlangıç değeri atamak için iki tamsayı kullanarak bir nokta nesnesi oluşturur.

Aşağıdaki kod parçası üç tip yapıcıyı kullanarak nokta oluşturur:

```
Point pt1 = new Point();
Point pt2 = new Point(new Size(10, 100));
Point pt3 = new Point(20, 200);
```

PointF yapısı Point yapısına benzerdir fakat tamsayı yerine kesirli nokta değerleri kullanır. Point yapısından farklı olarak Size veya SizeF nesnesi almaz. PointF sadece PointF() ve PointF(float,float) olmak üzere iki yapıcıya sahiptir.

Point ve PointF'in her ikisi de üç özelliğe sahiptir: IsEmpty, X ve Y. IsEmpty özelliği bir nokta boşsa yani X ve Y değerleri sıfır ise geriye "true", aksi halde "false"" döndürür. Yapıcılar Point() ve PointF() X ve Y değerleri sıfır olan bir boş alan oluşturur (Xu 2007).

### 3.1.10. Çizgiler (Lines) ve Eğriler (Curves)

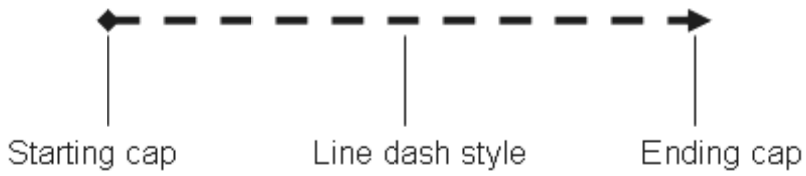
Line nesneleri düz çizgiler ve eğriler içerir. Düz bir çizgi koordinat çiftleri tarafından belirlenmiş noktaların DrawLine yöntemi kullanılarak birbirine bağlanmasıyla oluşturulur. Aşağıdaki kod parçası düz bir çizgi oluşturur.

```
Graphics g = new Graphics();
g.DrawLine(Pens.Black, x1, y1, x2, y2 );
g.DrawLine(Pens.Black, point1, point2);
```

(x1,y1) çizginin başlangıç noktası ve (x2,y2) çizginin bitiş noktasıdır. Başlangıç ve bitiş noktasını point1 ve point2 kullanarak doğrudan belirtmekte mümkündür. Koordinatlar veya noktalar tamsayı veya ondalıklı sayı olabilir. DrawCurve yöntemi bir nokta dizisindeki her bir noktadan geçen bir eğri oluşturur ve tüm noktaları düzleştirerek birbirine bağlar:

```
Graphics g = new Graphics();
g.DrawCurve(Pens.Black, new Point[ ]);
```

Bu durumlarda sadece düz çizgiler ve eğriler çizilir. Bununla beraber çizgiler ve eğriler birçok farklı biçimlerde olabilirler. Örneğin, Şekil 3.12'de gösterildiği gibi elmas başlangıç başlığı ile başlayan ve ok başlığı ile biten bir kesikli çizgi çizilebilir (Xu 2007).




**Şekil 3.12.** Başlangıç ve bitiş başlıklı Kesik çizgili bir çizgi (Xu 2007)

Bir çizgiye çizgi başlıklarını ve biçimlerini uygulamanın doğrudan yolu yoktur. Bu başlık ve biçimler bir Pen (kalem) sınıfı tarafından belirtilmelidir. Daha önce de bahsedildiği gibi bir çizgi çizmek için renk ve kalınlığı belirlenmiş bir kalem sınıfı kullanılmalıdır. Kalem sınıfı aynı zamanda çizgi başlıklarını ve çizgi biçimlerini ilişkilendirmek için üyelere (members) sahiptir. Kalem, bu belirli üyeleri belirterek farklı biçimlerdeki çizgileri çizmede kullanılabilir.

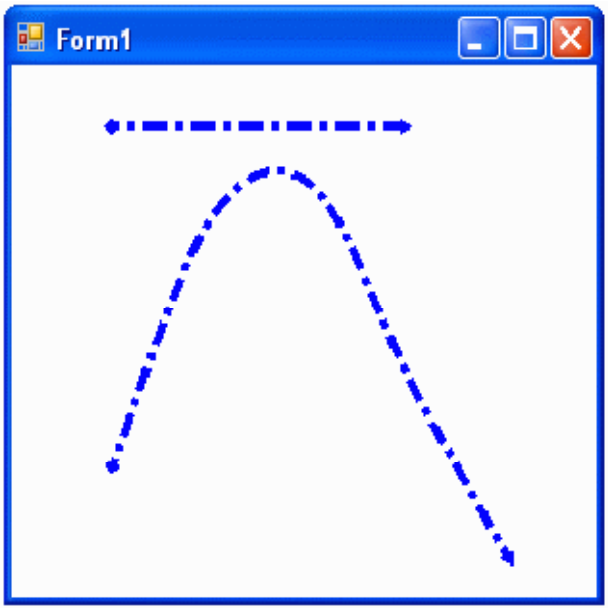
Bir örnekle farklı biçimlerde çizgi ve eğrilerin nasıl çizileceği gösterilecektir. Bir C# Windows Application oluşturuldu. Bu uygulamaya Example1_4 adı verildi. Aşağıda verilen bu projenin formunun kod listesi, renk ve kalınlığı tanımlanmış bir kalem

nesnesi oluşturur. Daha sonra kalem sınıfının StartCap ve EndCap özellikleri kullanılarak çizgi başlıkları, bunu takiben DashStyle ve DashOffset özellikleri ayarlandı. Son olarak çizgi ve eğrileri çizmek için DrawLine ve DrawCurve yöntemleri çağrıldı (Xu 2007).

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;
namespace Example1_4
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                        This.BackColor = Color.White;
                }
                protected override void OnPaint(PaintEventArgs e)
                {
                        Graphics g = e.Graphics;
                        // Bir kalem nesnesi oluştur.
                        Pen aPen = new Pen(Color.Blue, 4);
                        // Çizgi başlık ve biçimlerini ayarla.
                        aPen.StartCap = LineCap.DiamondAnchor;
                        aPen.EndCap = LineCap.ArrowAnchor;
                        aPen.DashStyle = DashStyle.DashDot;
                        aPen.DashOffset = 50;
                        //Düz çizgi çiz.
                        g.DrawLine(aPen, 50, 30, 200, 30);
                        // Eğri için nokta dizisi tanımla.
```

```
                Point point1 = new Point(50, 200);
                Point point2 = new Point(100, 75);
                Point point3 = new Point(150, 60);
                Point point4 = new Point(200, 160);
                Point point5 = new Point(250, 250);
                Point[] Points ={ point1, point2, point3, point4, point5};
                g.DrawCurve(aPen, Points);
                aPen.Dispose();
                g.Dispose();
            }
        }
}
```

Proje yapılıp çalıştırıldıktan sonra Şekil 3.13'te gösterilen sonuç elde edilir.




**Şekil 3.13.** Çizgi biçimli ve başlıklı düz çizgi ve eğri (Xu 2007)

### 3.1.11. Dikdörtgenler (Rectangles), Ellipses (Elipsler), ve Arcs (Yaylar)

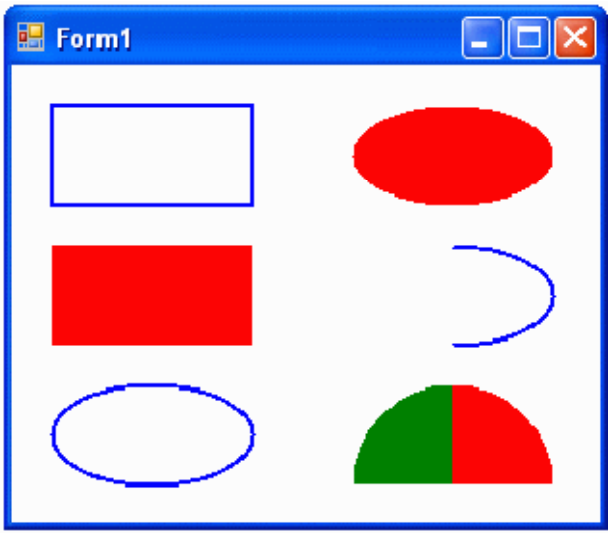
Rectangle ve RectangleF yapıları C# ve GDI+'da bir dikdörtgeni temsil eder. Bir Rectangle (dikdörtgen) yapısı bir dikdörtgen alanın yükseklik ve genişliğinin yanı sıra sol üst köşesini de saklar. Dikdörtgenin koordinatları olarak dört tamsayı ya da ondalık sayı değeri kullanarak ya da Point ve Size nesnelerinden bir dikdörtgen nesnesi oluşturulabilir. Rectangle ve RectangleF yapıları dikdörtgenin genişliği, yüksekliği ve pozisyonunu elde etmekte kullanılabilen özelliklere de sahiptir. Bir elips kendi sınırlayıcı dikdörtgeni tarafından tanımlanan dairesel bir şekildir. DrawEllipse yöntemi sınırlayıcı dikdörtgen içinde boş bir elips çizer. DrawEllipse yöntemi elipsin bütün sınırları etrafında kapalı bir eğri çizerken, DrawArc yöntemi elipsin bir kısmını çizer. Elipsin hangi kısmının çizileceği StartAngle ve SweepAngle ile belirtilir.

DrawPie yöntemi de elips ile yakından ilişkilidir. Bir elips kapalı dikdörtgen tarafından sağlanır. Buna ek olarak pastanın nerede başladığını ve elipsin ne kadar büyük olacağını tanımlamak için bir StartAngle ve bir SweepAngle tanımlanır. Visual C#'da temel grafik şekilleri kolayca oluşturulabilir. Yeni oluşturulan Example1_5'te aşağıdaki kodlar kullanılarak dikdörtgenler, elipsler, yaylar ve pastalar oluşturulabilir (Xu 2007).

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace Example1_5
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                        SetStyle(ControlStyles.ResizeRedraw, true);
```

```
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            // Create a pen object:
            Pen aPen = new Pen(Color.Blue, 2);
            // Create a brush object with a transparent red color:
            SolidBrush aBrush = new SolidBrush(Color.Red);
            // Draw a rectangle:
            g.DrawRectangle(aPen, 20, 20, 100, 50);
            // Draw a filled rectangle:
            g.FillRectangle(aBrush, 20, 90, 100, 50);
            // Draw ellipse:
            g.DrawEllipse(aPen, new Rectangle(20, 160, 100, 50));
            // Draw filled ellipse:
            g.FillEllipse(aBrush, new Rectangle(170, 20, 100, 50));
            // Draw arc:
            g.DrawArc(aPen, new Rectangle(170, 90, 100, 50),
            -90, 180);
            g.FillPie(aBrush, new Rectangle(170, 160, 100, 100),
            -90, 90);
            g.FillPie(Brushes.Green,
            new Rectangle(170, 160, 100, 100), -90, -90);
        }
    }
}
```

Bu proje Şekil 3.14'teki sonucu üretir.




**Şekil 3.14.** Proje Example1_5'ten elde edilen temel grafik şekilleri (Xu 2007).

## 3.2. USB-GPIB

### 3.2.1. Komutlar

#### 3.2.1.a GPIB'i başlatma

Herhangi bir GPIB kontrol programındaki ilk adım GPIB'i başlatmaktır. Bu "initialize" yordamı ile yapılır. Initialize yordamı iki bağımsız değişken ile çağrılır. Birincisi borda atamak istedeğin GPIB adresini verir ve ikincisi bordun sistem kontrolörü olup olmayacağını belirtir. Initialize yordamını çağırma işlemi aşağıdaki gibidir (QISI 2007):

**INITIALIZE (adres, düzey)**

**Adres,** bordunuz tarafından kullanılacak olan GPIB adresini veren, 0'dan 30'a kadar olan bir tam sayıdır.

**Düzey,** "0" sistem kontrolörünü ve "'2" cihaz modunu belirtir.

Initialize çağrıldığında birkaç işlem yapar. Birincisi, arayüzü işlem için hazırlar. İkincisi, bordunuz tarafından kullanılacak olan GPIB adresini ayarlar. Üçüncüsü, sistem kontrolü belirlenmişse GPIB sistemine girmek ve diğer cihazları başlatmak için arayüze IFC (interface clear-arayüz temizleme) gönderir (QISI 2007).

### 3.2.1.b. Bir cihaza veri gönderme

GPIB sistemi başlatıldıktan sonraki adım genellikle bir cihaza komut veya veri göndermektir. Send yordamı bunu kolaylaştırır.

**SEND (adres, bilgi, durum)**

**Adres,** verinin gönderileceği cihazın birincil adresine ayarlanan bir tam sayıdır (0-30 arasında).

**Bilgi,** cihaza gönderilecek olan stringtir (karakter dizisi). String gönderildiğinde sonlandırma karakteri otomatik olarak stringin sonuna eklenir. Varsayılan sonlandırma karakteri yeni satır karakteridir. Sonlandırma karakteri SetOutputEOS altyordamı ile değiştirilebilir.

**Durum,** iletimin tamam olup olmadığını gösterir. “0”, veri iletiminin tamam olduğunu “8”, zaman aşımı olduğunu yani cihazın cevap vermediğini gösterir.

Bu, “durum” argümanının ortaya çıktığı ilk seferdir. Genellikle tüm bordların çağırdığı en son argümandır. Durum iletim tamam olduğunda geriye sıfır değeri ile döner. Durum için sıfır olmayan en yaygın değer zaman aşımını gösteren sekizdir. Cihaz veriyi iletemediğinde veya veri iletimi zaman aşımı peryodundan daha uzun zaman aldığında bir zaman aşımı meydana gelir. Send yordamı sadece iki tane durum değerine sahiptir (QISI 2007).

### 3.2.1.c. GPIB adresleme

Birçok cihaz Send veya Enter gibi yordamların işleneceği, 0'dan 30'a kadar bir numara olan, sadece bir birincil GPIB adresine sahiptir. Örneğin;

```
send (16,"*IDN?",durum);
enter (r,80,&l,16,durum);
```

Bununla birlikte bazı cihazlar ikincil adresleriyle alt modüller içerirler. Bu cihazlara aşağıdaki formülde gösterildiği gibi birincil ve ikincil adresler birleştirilerek erişilebilir.

Formül : 100 * birincil + ikincil

Örneğin, birincil adres 5, ikincil adres 20 ise bu durumda 520 değerini kullanabilirsiniz.

```
send (520, "read?", durum);
```

Bu adres değerleri cihaz adres parametrelerini alan herhangi bir yordamla (send, enter, veya spoll gibi) kullanılabilir (QISI 2007).

### 3.2.1.d. Bir cihazdan veri okuma

Bir cihaz bir ölçüm aldığında sonraki adım bilgisayara veri okumaktır. "Enter" yordamı bir cihazdan okuma işlemi yapmak istediğinde kullanılır.

**ENTER (recv, maxlength, length, address, status)**

**Recv,** alınan veriyi içeren bir string değişkendir. Enter veri almayı;

- string full olduğunda,
- bir yeni satır karakteri alındığında,
- herhangi bir karakter EOI sinyali ile alındığında

sonlandıracaktır. Veriyle gelen satırbaşları (yeni satır karakteri) engellenir ve alınan veriye konmaz.

**Maxlength,** almak istediğiniz maksimum karakter sayısını belirten bir değerdir. Maxlength 0'dan 65535 (heksedesimal FFFF)'e kadar bir sayı olabilir.
**Length,** alınan karakterlerin gerçek sayısını içerir.
**Address,** okunacak olan cihazın GPIB adresidir.
**Status,** iletimin tamamlanıp tamamlanmadığını (0=tamam, 8=zaman aşımı) gösterir.

Enter yordamı işlendikten sonra, string değişkeni alınan veriyi içerir. Alınan verinin baytlarının uzunluğu length argümanında geri dönecektir.

Aşağıda veri elde etmek için initialize, enter ve send yordamları kullanılarak C#'da yazılmış örnek bir program verilmiştir (QISI 2007).

```
Private void button1_Click(object sender, System.eventArgs e)
{
int status,len;
String r;
CEC488.initialize (21,0);                          // GPIB sistemini başlat
CEC488.send (7,"MEASURE", out status);     // Ölçme yapmak için bir komut gönder
CEC488.enter ( out r,30, out len,7, out status); // Ölçüm değerini oku.
textBox1.Text = "Değer:  " + r; //Değeri textBox1'e yazdır.
}
```

### 3.2.1.e. Cihaz durumunu kontrol etmek (servis isteği)

GPIB cihazları seri sorgulama ve paralel sorgulama olarak bilinen iki teknik sayesinde durum bilgisini ele edebilirler. Cihazların ihtiyacı olan bazı eylemleri sinyal vermesi için SRQ (service request-servis isteği) olarak adlandırılan bir sinyal de vardır.

SRQ fonksiyonu ile servis istekleri için test yapılabilir. Bu fonksiyon herhangi bir cihaz servis istediğinde geriye "true" değerini döndürür. Hangi cihazın servis isteğinde bulunduğuna karar vermek için seri sorgu yordamı (spoll) kullanılabilir (QISI 2007).

**- Seri sorgulama**

Bir seri sorgulama tek bir cihazın durumunu okur.

**SPOLL (address,poll,status)**

**Adress,** sorgulanacak olan cihazın GPIB adresidir.
**Poll,** sorgunun sonuç baytını içerecektir.
**Status,** sorgunun tamam olup olmadığını gösterir.
"0" ise tamam,
"8" ise zaman aşımı (cihaz cevap vermiyor).

Bir cihazdan seri sorgulama durumu genelde her biri "ölçüm tamamlandı", "hata" veya "aralık dışı" gibi cihaz bağımlı mesajlarına sahip olabilen bir 8 bit grup olarak değerlendirilir. En soldaki bitin yanındaki bit cihazın servis isteğinin (SRQ hattında) olup olmadığını göstermek için ayrılmıştır (QISI 2007).

**- Paralel sorgulama**

**PPOLL (poll)**

Paralel sorgulama cihaz durumunu belirlemede kullanışlı olabilir. Bir paralel sorgulama her bir biti bir cihazın servis isteğini gösterebilen bir tek baytı geri döndürür. Bu şekilde ayrı ayrı cihazların tek tek sorgulanmasına gerek kalmaz.

**PPOLL (poll)**

**Poll,** sorgulama değeriyle(0'dan 255) geri döner.

Ppoll'un status geri dönüş argümanı yoktur. Çünkü zaman aşımına uğramaz. Sorgulama cevap değerinin her bir biti bir veya daha fazla cihazın servis isteğini gösterebilir. Cihazlar, içlerinden ayarlanarak ya da ayarlama komutlarının paralel sorgusuyla belirli bitlere atanabilir (QISI 2007).

### 3.2.1.f. Bir cihazın hazır olması durumunu test etme

ListenerPresent bir dinleyiciye veri göndermeye başlamadan önce, onun GPIB'de gerçekten hazır olup olmadığını test etmek için çok kullanışlıdır. Programın başında kullanıcı cihaz cevap vermiyor ve bağlantı kopmuş ya da kapanmış olabilir diye uyarılabilir. IEEE-488 arayüzünün bazı modelleri dinleyiciye herhangi bir veri göndermeden onun hazır olduğunu fark eden özel donanımlara sahiptir.

Bu özelliğin donanım tarafından desteklenip desteklenmediğini kontrol etmek için, GPIBFeature() yordamı çağrılır. Özelliğin desteklendiğini bildiğinde, listener present yordamı çağrılabilir (QISI 2007).

**ListenerPresent (addr)**

**Addr,** kontrol etmek istediğimiz cihazın adresi. Sıfır ise cihaz hazır değil, sıfır değil ise cihaz hazırdır.

Eğer senin donanımın bu özelliği desteklemiyorsa, yine de bağlantısı olmayan cihazları bulabilirsin. Çünkü "Send" yordamı herhangi bir cihaz hazır olmadığında, tam zaman aşımı olmasını beklemeden, hemen status değerini "8" geri döndürecektir (QISI 2007).

### 3.2.1.g. GPIB bordunun hazır olması durumunu test etme

Cihaz komutlarını göndermeye başlamadan önce bilgisayarda gerçekten bir bordun hazır olup olmadığını test etmek için kullanışlı bulabilirsiniz.

**GPIBBordPresent**

Geriye sıfır döndürdüğünde bord hazır değil demektir. Eğer bord yüklüyse sıfır olmayan bir değeri geri döndürür.

(Not: Geriye doğru uyumluluk için, arayüz bordlarının bazı farklı modellerinde bu yordam halen farklı kodlar geri döndürür. Fakat bu bordun işlevselliğini belirlemek için doğru bir yol değildir) (QISI 2007).

### 3.2.1.h. Donanım bordunun GPIB özelliklerini kontrol etme

Bir programda bordun ayarlarını ve özelliklerini kontrol etmek için kullanışlı olabilir.

**GPIBFeature (feature)**

**Feature,** elde etmek istediğimiz bilgiyi gösteren aşağıdaki listeden bir numaradır.

**Çizelge 2.12.** GPIBFeature sorgulama sayıları ve anlamları (QISI 2007)

| Feature | Number | Description |
|---|---|---|
| IEEEListener | 0 | Bord ListenerPresent fonksiyonunu destekler mi? Eğer desteklerse, ListenerPresent daima"true" geri |
| IEEE488SD | 1 | Bord 488SD yüksek hız protokolünü destekler mi? |
| IEEEDMA | 2 | Bord DMA donanımını kullanır mı? |
| IEEEFIFO | 3 | Donanım yüksek hız iletimini desteklemek için bir FIFO tamponuna sahip midir? |
| IEEEIOBASE | 100 | Temel I/O adresini geri döndürür. |
| IEEETIMEOUT | 200 | Geçerli zaman aşımı ayarlarını geri döndürür. |
| IEEEINPUTEOS | 201 | Geçerli giriş EOS'i geri döndürür. |
| IEEEOUTPUTEOS1 | 202 | Geçerli çıkış EOS1'i geri döndürür. |
| IEEEOUTPUTEOS2 | 203 | Geçerli çıkış EOS2'yi geri döndürür. |
| IEEEBORDSELECT | 204 | Geçerli bord numarasını geri döndürür. |
| IEEEDMACHANNEL | 205 | Geçerli DMA kanalını geri döndürür. |

Kullanılan arayüz bordu hakkında arzu edilen bilgiyi geri döndürür. Evet/Hayır özellik sorgusu için bir sıfır ya da sıfır olmayan değer geri döndürür.

### 3.2.1.i. Send ve Enter'ın kullanımındaki kısıtlamalar

Send ve Enter veri iletmek ve almak için basit yordamlardır. Onlar birçok cihazın üstesinden gelmek için dikkatlice tasarlanmıştır fakat uygulamanda geçerli olmayabilecek bazı varsayımlarda bulunurlar. Özellikle,

Bordunuz sistem kontrolörü olmalı. Send ve Enter bordunuzu GPIB kontrolörü varsaymalıdır. Eğer başka kontrolör varsa ve bordunuz bir cihaz olarak kullanılıyorsa gelişmiş programlama kısmı okunmalıdır.

Cihaz bir yeni satır veya EOI kabul etmeli ve veri sonlandırıcı olarak bir yeni satır veya EOI göndermelidir. Enter bir yeni satır veya EOI aldıktan sonra veri almayı sonlandırır ve Send iletilen stringin sonuna EOI ile başlayan yeni satır ekler. Birçok cihaz bunu ister. Ancak eğer cihanızın başka gereksinimleri varsa Transmit ve Receive yordamlarını kullanmanız gerekecektir. Cihaz içerisinde yeni satırları barındıran ikilik veri yerine karakter verisini iletir. İkilik veri için Tarray ve Rarray yordamları önerilir (QISI 2007).

### 3.2.1.j. Transmit kullanarak düşük seviye kontrol

IEEE-488 standardı basit veri transferine ek olarak birkaç özel amaçlı komut tanımlar. Transmit komutu GPIB üzerinde herhangi bir komut ve uygulamayı Send komutu ile sağlanandan daha ince seviyede kontrol edebilen programlama becerisi kazandırır (QISI 2007).

**TRANSMIT (command,status)**

**Command,** bir grup GPIB komutları ve verileri içeren stringtir. Her bir parçası bir veya daha fazla boşlukla ayrılmıştır. Takip eden kısımda uygun komutların tam bir listesi verilmiştir.

**Status,** komutların gönderilip gönderilmediğini gösterir.

0 – Tamam
1 – Aykırı komut sözdizimi
2 – Bir konuşmacı yokken veri gönderilmeye çalışıldı
4 – Bir konuşma veya dinleme listesinde aktarılmış string veya END komutu bulundu
8 – Dinleyen cihaz yok veya zaman aşımı
16 – Bilinmeyen komut

Durum değeri Transmit için biraz daha karışıktır. 0'dan 31'e kadar herhangi bir değer olabilir. Her bir bit belirli tip hatayı gösterir. Bir sıfır değeri her zamanki gibi iletimin tamamlandığını gösterir. Sekiz değeri komutla alakalı yanlış olmadığını fakat cihazın cevap vermediğini bildiren bir zaman aşımını gösterir. Diğer değerler tipik olarak komut stringlerini yazımdaki hatalardan kaynaklanıyor. Bir örnek olarak, eğer status 24, 26 veya 8 ise bunun anlamı "stringte bilinmeyen komut bulundu ve aynı zamanda bir zaman aşımı oluştu" demektir.

Aşağıda gösterilen örnekler iki aygıtı aynı anda veri alması için programlar ve daha sonra onların ölçümlerini Group Execute Trigger komutu kullanarak senkronize eder (QISI 2007).

Komut stringi uygulanmış bir grup GPIB komut olarak Transmit yordamı tarafından yorumlanır. Bu durumda UNL (unlisten) var olan dinleyicileri pasif eder. Dizi "LISTEN 4 7" cihazlara 4 ve 7 adreslerini dinleyici olarak atar. Son olarak "GET" Group Execute Trigger komutunu belirtir.

transmit("UNL LISTEN 4 7 GET", status);

### 3.2.1.k. Tranmit komutları – Veri iletimi

**LISTEN**

LISTEN bir veya daha fazla dinleyici tanımlar. LISTEN cihazın GPIB adresini gösteren bir dizi numara tarafından takip edilmelidir (QISI 2007).

Örneğin:
"LISTEN 1"
"LISTEN 4 9 30"

**TALK**

TALK bir konuşmacı tanımlar. Aynı anda sadece bir konuşmacı olabilir. Eğer birçok konuşma komutu verildiyse, listenin sonundaki komut etkisini gösterir (QISI 2007).

Örneğin:
"TALK 3"
"TALK 9"

**SEC**

SEC bir ikincil adres tanımlar. Bu komut bir sayı tarafından takip edilmelidir. SEC LISTEN veya TALK ile gönderilen cihazın birincil adresinden sonra kullanılır (QISI 2007).

Örneğin:
"LISTEN 4 SEC 8"
"TALK 8 SEC 9"

**UNT**

Untalk. O anki konuşmacıyı veya herhangi birini devre dışı bırakır.

**UNL**

Unlisten. Dinleyici olarak atanmış birini devre dışı bırakır.

**MTA**

My Talk Address (konuşma adresim). Bordunuzu konuşmacı olarak atar.

**MLA**

My Listen Address (dinleme adresim). Bordunuzu bir dinleyici olarak atar.

**DATA**

Tüm dinleyicilere iletilmesi gereken veriyi gösterir. Bu komutu göndermeden önce bilgisayar bir konuşmacı olarak atanmalıdır. Veri iki şekilde gösterilebilir: Tek tırnak(‘) içerisine alınmış bir string olarak veya 0’dan 255’e kadar bir sayı olarak. Tırnak içerisine alınmış stringler karakter olarak gönderilir. Sayılar veri olarak gönderilmek üzere bir bayt değeri gösterir. Onlar satır başı (13) ve yeni satır (10) gibi yazdırılmayan karakterlerin gönderilmesinde kullanışlıdır (QISI 2007).

**END**

Sonlandırma bayt(lar)ını (varsayılan EOI sinyaliyle yeni satırdır) gönderir. Bu sadece DATA komutundan sonra kullanılmalıdır.

Yukarıda gösterilen komutlar grubu tüm veri iletimini gerçekleştirecektir. Takip eden diğer komutlar GPIB kontrolü için özelleştirilmiştir (QISI 2007).

### 3.2.1.l. Transmit örnekleri

**"UNL UNT LISTEN 4 MTA DATA 'merhaba' END"**

Bu komut tüm dinleyicileri ve konuşmacıları kapatır (UNL UNT), daha sonra cihaz 4'ü bir dinleyici olarak, bilgisayarı bir konuşmacı olarak atar (MTA) ve cihaz 4'e string veri olarak 'merhaba' gönderir. Son olarak EOI ile yeni satır gönderilir (END).

**"DATA 'testing' 13 10"**

Bu komut bilgisayarın bir konuşmacı olduğunu ve bir veya birden fazla cihazın dinleyici olduğunu varsayar. Veri stringi "testing" ardından bir satır başı ve yeni satır ile gönderildi (QISI 2007).

**"DATA 27 '&k2S'"**

Bu veri dizisi Escape(ASCII 27) ardından '&k2S' gönderir.

**"UNL LISTEN 4 8 DATA 'info' 13 10 'second line' 13 10"**

Bu komut tüm dinleyicileri kapatır, daha sonra cihaz 4 ve 8'i dinleyici olarak atar, daha sonra bu cihazlara her bir satırı yeni satır ve satır başı ile biten iki satır gönderir.

### 3.2.1.m. Ek veri iletim komutları

**REN**

Remote Enable (Uzak Etkinleştirme). Bu komut uzak etkinleştirme sinyalini açar. Sadece sistem kontrolörü bu sinyali sağlayabilir. Bazı cihazlar komutları kabul etmeden önce Remote Enable gerekir.

**EOI**

End or Identify (Bitir veya Kimlik Tanımla). Bu komut veri komutu gibi işler fakat bir iletimin son baytı olduğunu göstermek için veri baytını EOI sinyali ile gönderir. Aşağıdaki örneğe inceleyin.

**GTL**

Go To Local. O anki dinleyicilere işlemlerine önlerindeki panelden devam etmesini söyler (bilgisayar tarafından uzaktan kontrole engel olur). Bu komut aynı zamanda uzak etkinleştirme sinyalini (remote enable signal) kapatır (QISI 2007).

### 3.2.1.n. Örnekler

**"MTA LISTEN 4 REN DATA 'hello' 13 10"**

Bu komut adres 4'teki cihaza 'hello' ardından yeni satır ve satır başı gönderecektir. Uzak etkinleştirme (Remote enable) veri gönderilmeden önce açılacaktır. Not: Uzak

etkinleştirme Initialize çağrılana veya GTL komutu kullanılana kadar açık kalacaktır (QISI 2007).

**"DATA 'hello' EOI 10"**

Bu komut ‘hello’ ardından EOI sinyaliyle yeni bir satır gönderir. ( ‘EOI 10’ ‘END’e eşittir).

**"UNL LISTEN 5 8 GTL"**

Bu komut, cihaz 5 ve 8’i dinleyici olarak ayarlar ve daha sonra onlara işleme ön panelden devam etmelerini söyler.

### 3.2.1.o. Seri sorgulama ayarları

**SPE**

Serial poll enable (seri sorgulama aktif). Bu komut bir cihazdan seri sorgulama cevabını elde etmek için kullanılır. SPOLL yordamı ile birlikte kullanılır. Bir cihaz konuşmaya adreslendiğinde ve SPE komutu aldığında, normal veri yerine seri sorgulama yanıtını gönderecektir.

**SPD**

Serial poll disable (seri sorgulama pasif). Bu komut sorgulanan cihazı geri normal konuşmacı durumuna koyar.

Örnek; **"UNL MLA TALK 5 SPE"**

Bu komut adres 5'teki cihazı bir seri sorgulama yanıtı elde etmek için kurar.

### 3.2.1.ö. Paralel sorgulama ayarları

**PPC**

Parallel poll conŞekil (paralel sorgulama ayarla). Bu komut o andaki dinleyici(ler)e takip eden paralel sorgu etkinleştirme komutunu beklemesini söyler.

Bir paralel sorgu etkinleştirme komutu 96 ve 111 arasında bir GPIB komut baytıdır (CMD komutu ile gönderilir). Bu komut bir ikilik (binary) bayt: 0110SPPP şeklinde değerlendirilir. Burada "S" cihaz tarafından servis isteği (0 veya 1) için kullanılacak olan bit değerini ve "PPP" yanıt için kullanılacak olan bit sayısını gösteren, 0'dan 7'e kadar bir ikilik değeri belirtir.

Örneğin, bir 105 paralel sorgu etkinleştirme komutu ikilik 01101001'dir. Son 001, kullanılacak olan bit sayısı 1 demektir ve önce gelen 1 cihaz servis isteği olduğunda bu bite yerleştirilecek "1" veya "true" değerini gösterir (QISI 2007).

**PPD**

Parallel poll disable (paralel sorgu pasif). Bu komut PPC komutunu da takip edebilir. Ayarlanmış cihaz için herhangi bir paralel sorgu yanıtını devre dışı bırakır (QISI 2007).

**PPU**

Parallel poll unconfigure (paralel sorgu pasif etme). Bu komut tüm cihazlardaki (dinlemeye adreslenmiş veya adreslenmemiş) paralel sorgu yanıtlarını pasif eder.

**Örnekler**

**"PPU"**

Bu komut tüm paralel sorgulama yanıtlarını devre dışı bırakır.

**"UNL LISTEN 1 2 PPC PPD"**

Bu komut GPIB adresi 1 ve 2 olan cihazların paralel sorgu yanıtlarını devre dışı bırakır.

**"UNL LISTEN 5 PPC CMD 105"**

Bu komut GPIB adresi 5 olan cihazın paralel sorgu yanıtını bir bit sayısındaki değeri "1" olması için aktif eder.

**3.2.1.p. Diğer çoklu satır komutları**

**DCL**

Device Clear (cihaz temizleme). Tüm cihazlara önceden tanımlı durumlara ayarlanmalarını söyler. Bu komutu aldıktan sonra cihazlar tarafından yapılan işlem cihaz bağımlıdır (QISI 2007).

**LLO**

Local Lockout (yerel kilitlenmiş). Cihazların yerel olarak ön panelden kontrolünü devre dışı bırakır. Bu komut tamamen kontrolün bilgisayar tarafından yapılmasını garanti etmek için genellikle REN ile birlikte kullanılır ve cihazlar üzerindeki ön panelin kullanımına engel olur. Tüm cihazlar bu komutu uygulayamaz.

**CMD**

Command (komut). Bu DATA komutuna benzer şekilde çalışır, CMD'yi takip eden bilgiler hariç ATN hattı ile gönderilir, baytları GPIB komutu yapar. Bu sayede standart olmasalar bile herhangi GPIB komut baytının gönderilmesine izin verir. CMD'nin bir ortak kullanımı bir paralel sorgu etkinleştirme komutu göndermektir (QISI 2007).

### 3.2.1.r. Diğer adresleme komutları

**GET**

Group Execute Trigger (Grup Yürütme Başlatmak). O an dinlemeye adreslenmiş tüm cihazların bir cihaz bağımlı işleme (genellikle bir ölçüme) başlamasına sebep olan bir GPIB komutudur. Bu komut birçok cihazı senkronize etmeye çalışırken kullanışlıdır. Bazı GPIB cihazları bu komutu uygulayamaz.

**SDC**

Selected Device Clear (Seçili Cihazı Temizle). Bu komut DCL'ye benzer fakat sadece o an dinlemeye adreslenmiş cihazları resetler.

**TCT**

Take control (Kontrol al). Bu komut kontrolör tarafından diğer cihazın kontrolör kapasitesi ile GPIB üzerindeki kontrolü almasına izin verir (QISI 2007).

### 3.2.1.s. Diğer komutlar

**- IFC:** Interface Clear (Arayüz Temizleme). Bu komut sadece sistem kontrolörü tarafından gönderilebilir. GPIB sistemi üzerindeki tüm cihazların arayüz durumunu resetler. Bu komuttan sonra, dinlemeye veya konuşmaya adreslenmiş cihaz kalmayacaktır. Eğer kontrol diğer cihaza geçtiyse, sistem kontrolörü GPIB kontrolünü tekrar kazanacaktır. Not: Initialize yordamı bu komutu içten kullanır (QISI 2007).

**- Bir cihaz temizleme komutu gönderme**

Transmit yordamı bir cihaz temizleme gönderimi için kullanılabilir. İki tip cihaz temizleme vardır: evrensel temizleme ve seçili cihazı temizleme. Evrensel temizleme göndermek için bu Transmit komut stringi kullanılır:

**"DCL"**

Bir seçili cihaza temizleme göndermek için istenilen cihazlar dinlemeye adreslenir ve daha sonra bir SDC kullanılır. Örneğin, cihaz 6'yı temizlemek için şu Transmit komut stringi kullanılabilir:

**"UNL LISTEN 6 SDC"**

**- Bir cihaz başlatma komutu gönderme**

Transmit yordamı bir cihaz başlatma komutu göndermek için kullanılabilir. Başta istenilen cihaz(lar) dinlemeye adreslenir ve daha sonra bir Group Execute Trigger (GET) kullanılır. Örneğin, cihaz 9'u başlatmak için şu Transmit komut stringi kullanılır:

**"UNL LISTEN 9 GET"**

### 3.2.1.ş. Receive yordamı

Receive bir cihazdan veri okumak için kullanılabilir. Enter'a benzer fakat GPIB üzerinde bir konuşmacı adreslemez ya da PC'yi bir dinleyici olarak tanıtmaz. Bu yüzden Transmit ile kullanılmalıdır.

**Receive (recv, maxlength, length, status)**

**Recv,** alınan veriyi içerecek olan bir string değişkendir. Recv en azından almak istediğiniz karakter kadar karakter içeren bir string olarak başlatılmalıdır.

Receive veri almayı; string tam dolduğunda, yeni bir satır alındığında veya EOI sinyaliyle herhangi bir karakter alındığında sona erdirecektir. Gelen veri içindeki satır başları göz ardı edilir ve recv içine konmaz.

**Maxlength** almak istenilen maksimum karakter sayısını belirleyen bir değerdir. Makslength 0'dan 65535 (hex FFFF)'e kadar bir sayı olabilir.
**Length** alınan karakterlerin gerçek sayısını içerecektir.
**Status** iletimin tamamlanıp tamamlanmadığını gösterir.

0 – Tamam
2 – PC bir dinleyici değilken almaya çalışıldı
8 – Zaman aşımı

Receive, Enter yordamının kullanılmadığı durumlarda kullanışlıdır. Uzun stringlerin receive yordamı tekrar tekrar çağrılarak parçalar halinde alınmasında veya bilgisayarın GPIB kontrolörü olmadığı durumlarda kullanımı söz konusudur.

Not: Eğer Receive boyunca bir zaman aşımı meydana gelirse, bu muhtemelen bazı verilerin halen okunmasından dolayıdır. Cihaz bazı karakterleri gönderir, daha sonra

devam etmeden önce uzun zaman ara verirse bu meydana gelebilir. Bu durumda, length sıfır olmayacak ve status sekiz olacaktır. Verinin kalanını almaya devam etmek için sadece Receive komutunun tekrar çağrılması yeterlidir (QISI 2007).

### 3.2.1.t. İkilik (Binary) veri iletimi

Bazı cihazlar veriyi ikilik formatta gönderir veya kabul eder. Send, Enter ve Receive yordamları ikilik veri için uygun değillerdir. Çünkü onlar özel ASCII karakterler gibi belirli bit örneklerini değerlendirirler. Send bir yeni satır ekler veriye ve Enter ve Receieve bir yeni satır aldıklarında alma işlemini sonlandırır.

Tarray ve Rarray yordamları ikilik verileri işlemek için verilmiştir. Ayrıca, bu rutinler veri iletimini daha hızlı sağlamak için en iyi şekilde tasarlanmışlardır. Bu yordamların hiçbiri GPIB adresleme yapmaz, bu yüzden onları çağırmadan önce Transmit ile konuşmacı ve dinleyicileri ayarlamak gerekir (QISI 2007).

**- Tarray**

Tarray bilgisayardan o anki dinleyici cihazlar grubuna bir uzun ikilik veri bloğu göndermemize izin verir. EOI sinyali isteğe bağlı olarak son veriyle birlikte gönderilebilir.

**TARRAY (data.array, count, eoi, status)**

Data.array iletilecek olan bilgidir.
Count iletilecek olan baytların sayısıdır.
EOI son veri baytı ile birlikte EOI'nın gönderilip gönderilmeyeceğini gösterir.
0 – Hayır
1 – Evet

Status iletimin tamam olup olmadığını gösterir.
0 – Tamam
2 – PC bir konuşmacı değilken veri gönderilmeye çalışıldı.
8 – Zaman aşımı
Tarray bilgisayar hafızasında bulunan baytları olduğu gibi gönderir.

**- Rarray**

Rarray yordamı 64K bayta kadar ikilik veri almak için kullanılır. Verilen bayt sayısına ulaşıldığında veya EOI sinyali ile bir bayt alındığında veri almayı sonlandırır (QISI 2007).

**RARRAY (data.array, count, length, status)**

Dara.array alınan veriyi içerek olan dizidir.
Count alınacak olan baytların sayısıdır.
Length alınan baytların gerçek sayısıdır.
Status iletimin tamam olup olmadığını gösterir.
0 – Tamam
2 – PC bir dinleyici değilken veri almaya çalışıldı.
8 – Zaman aşımı
32 – EOI sinyali ile biten başarılı iletim

Not: Rarray tarafından geri döndürülen status değeri iletim tamam olsa bile sıfır olmayabilir. Rarray iletim EOI sinyali ile sonlandırıldığında 32 durumunu geri döndürür.

### 3.2.1.u. Veri formatları

Çoğu durumda ikilik veri için veri dizilerini bir bayt, tamsayı veya gerçek tip değişken olarak bildirmeliyiz. Seçtiğimiz tip, cihaz tarafından gönderilen veri formatına bağlıdır.

Bazen cihaz tarafından gönderilen veri, PC veya yazdığımız dil tarafından kolayca kullanılacak formatta değildir. Bir yaygın durum bir cihaz 16 bit veri gönderir fakat baytlar PC'nin depolayacağı şeklin tersi şekilde sıralanmıştır. Diğer yaygın durum bir cihaz 4 bayt kayan noktalı veri gönderir fakat yine baytlar tersten sıralanmıştır.

Bu durumlarda verilere tek bayt olarak erişip onları tekrar biçimlendirmeliyiz. Böylece bilgisayar onları işleyebilir. ASCII karakter gibi diğer formatlardaki iletim verilerine sahip cihazları veri format dönüştürme probleminden sakınmak için araştırmak isteyebiliriz (QISI 2007).

### 3.2.1.ü. Yüksek hız veri iletimi

IEEE-488 bordlarımız yüksek hız iletim için tüm donanımı kapsar. En iyi veri hızına ulaşmak için Tarray ve Rarray yordamları kullanılır. Diğer yordamlar (Send ve Enter) rahat kullanım ve çoğunlukla daha kısa yazı stringleri için tasarlanmıştır. Tarray ve Rarray her bordun yüksek hız donanım özelliklerini kullanmak için en iyi şekilde ayarlanmıştır. İletim hızı genellikle GPIB cihazları tarafından kısıtlanır. Birçok cihaz maksimum 100Kbayt/saniye ya da daha az hıza sahiptir. Tüm bordlarımız bu hızın birkaç katı hızda veri iletebilir.

**- 16-bit ISA Yol Arayüzü (488EX)**

488EX 1Mbayt/saniye'nin üzerinde hıza ulaşmak için 16 bit I/O işlemleri ve özel donanım kullanır. Tarray ve Rarray yordamları otomatik olarak 488EX donanımı kullanır ve herhangi bir ek yordamı çağırmaya gerek yoktur. Bununla beraber eğer

488EX veya diğer bordların birinde çalışacak bir program yazıyorsanız bir DmaChannel çağrısı içermesini isteyebilirsiniz. Bu çağrının 488EX'te bir etkisi yoktur fakat diğer bordların daha hızlı çalışmasına izin verecektir (QISI 2007).

**- PCI Yol Arayüzü**

PCI488, DMA kullanmaz, bu yüzden DmaChannel çağrısı bu bord için engellenmiştir. Diğer donanım yöntemleri bu bordda optimum iletim hızı için kullanılır.

**- 8-bit ISA interface**

PC488, FIFO tamponlarını kullanır ve sadece tarray ve rarray'i çağırmaya ihtiyacımız vardır. Bu bord için Windows altında dmachannel() çağırmanın etkisi yoktur. DMA'ya ihtiyaç olmadan son iletim hızına ulaşılır.

**- DOS'ta diğer ISA yol arayüz methodları**

Bu bordlar bilgisayar içinde yerleştirilmiş direct memory access – direk hafıza erişimini (DMA) kullanır. PC488, DMA kanalları 1 veya 3'ü kullanabilir. 4x488, DMA kanal 1'i kullanır. PS488 bir mikro kanal bilgisayarında DMA kanallarını 0'dan 7'e kadar kullanabilir.

Not: Diğer ekli bordlar da DMA'yı kullanır ve bir kanalın diğer bordlar ile çakışmaması için seçilmiş olması gerekir. Kanal 1 genelde uygundur.

**- DMAChannel**

DMA başlangıç olarak pasiftir. DMA'nın etkinleştirilmesi için DmaChannel yordamı çağrılır.

**DMACHANNEL (channel)** :

Channel arayüz bordu tarafından kullanılacak kanal numarasını gösterir. Channel bordun donanım yapılandırma ayarlarını eşleştirmelidir. DMA'yı tekrar pasif etmek için DmaChannel kanal değeri 1 ile çağrılmalıdır.

Not: Bazı donanım modellerinde bu çağrı engellenmiştir. Çünkü donanım DMA iletimlerini desteklemez ya da kullanmaz (QISI 2007).

### 3.2.1.v. Bord (Board) parametrelerini yapılandırma

Bir uygulama standart olmayan bir donanımın kurulumuna veya standart olmayan arayüz ayarlarına ihtiyaç duyduğunda, bu değerler bu bölümde tanımlanmış yordamlarla değiştirilebilir. Çoğunlukla bu yordamlara ihtiyaç kalmaz.

**SetPort**

**SETPORT (bord, port)**

Bord 0'dan 3'e kadar bord numarasıdır. Sadece bir IEEE-488 arayüz borduna sahipseniz, kullanılan bord numarası 0'dır.

Port bordun I/O adresidir. Varsayılan fabrika ayarı 2B8 Hex'dir.

SetPort yordamı, arayüz bordu I/O adresi fabrika ayarı 2B8 Hex'ten başka bir adrese ayarlandığında kullanılır. Bilgisayardaki başka bir donanımla çakışmadan dolayı ya da birçok bord arayüzü kullanıldığında bu meydana gelebilir.

Not: I/O port adresi otomatik bilindiğinden beri PCI yol bordları gibi tak ve çalıştır bordları için bu çağrı göz ardı edilecektir.

Not: Programda SetPort() çağrılmasa bile uygulanan bordlar üzerinde varsayılmayan adresleri ayarlamak için kullanabilir (QISI 2007).

### 3.2.1.y. Bord (Board) seçimi

**BORDSELECT (bord)**

Bord 0'dan 3'e kadar arayüz bord numarasıdır. BordSelect sadece çoklu GPIB arayüzü bilgisayara yüklendiğinde kullanılır.

### 3.2.1.z. Zaman aşımını ayarlama

**- SETTIMEOUT (msec)**

Msec milisaniye cinsinden yeni zaman aşımı değeridir. Zaman aşım peryodu bir hata bildirilmeden önce giriş veya çıkış baytları arasındaki izin verilen maksimum zamandır. Yüksek hız iletimi için DMA kullanıldığında zaman periyot aşımı baytlar arasındaki zaman değil, tüm iletime uygulanır. Msec 55 milisaniyenin en yakın katına yuvarlanmış olabilir.

Varsayılan zaman periyodu 10 saniyedir (10000 msec). SetTimeot eğer varsayılan zaman peryodu olan 10 saniye uygulamanız için uygun değilse kullanılır. Eğer çok yavaş bir cihazınız varsa, zaman aşımını uzatmak gerekebilir. Eğer hızlı bir cihazınız varsa ve hataların 10 saniyeden daha az sürede bildirilmesini istiyorsanız zaman aşımını kısaltabilirsiniz (QISI 2007).

**- SetOutputEOS**

**SETOUTPUTEOS (eos1,eos2)**

eos1 ve eos2 Transmit'te END komutu kullanıldığında veya Send yordamının sonunda gönderilecek olan sonlandırma karakterleridir. Eğer eos2 bir boş karakterse (0) sadece bir karakter gönderilir.

Varsayılan output end of string (OutputEOS) bir yeni satırdır.

SetOutputEOS, varsayılandan başka bir sonlandırma karakterine sahip cihazınız varsa kullanılır. Bazı cihazlar hem bir satır başı hem de bir yeni satıra ihtiyaç duyar. Transmit yordamı kullanılarak gönderilen veri baytları üzerinde tam kontrol elde edilebilir.

**- SetInputEOS**

**SETINPUTEOS (eos)**

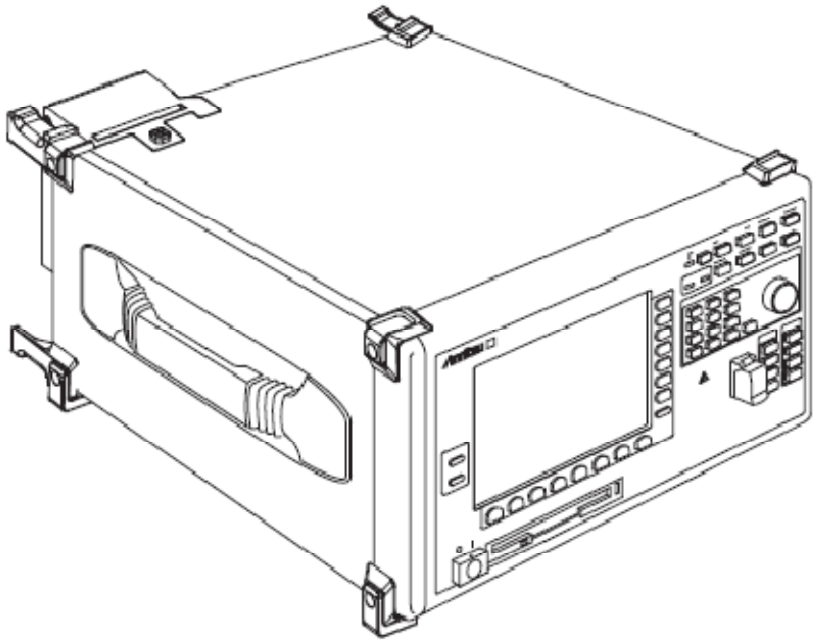
eos Enter ve Receive yordamları tarafından kullanılan sonlandırma karakteridir. Eğer eos yeni satırsa (10), o zaman satır başı karakterleri giriş stringinden silinir.

Varsayılan input end of string (InputEOS) bir yeni satırdır.

SetInputEOS varsayılan yeni satırdan başka bir karakterle iletişimini sonlandıran bir cihazınız varsa kullanılır. Not: Enter ve Receive önceden tanımlanmış bir karakter sayısı aldıklarında veya GPIB EOI sinyali alındığında da alma işlemini sonlandırır (QISI 2007).

## 3.3. Anritsu MS9710B Optik Spektrum Analizörü

### 3.3.1. Tanıtım



**Şekil 3.15.** Anritsu MS9710B Optik Spektrum Analizörün bir görünüşü (Anritsu 2008)

600'den 1750 nm'ye kadar olan dalgaboyu aralığındaki optik seviyeler maksimum 0,07 nm çözünürlük ile ölçülebilir. Seviye ölçüm aralığı -90 dBm'den +10 dBm'e kadardır ve harici zayıflatıcı on yapılarak 20 dBm kadar değer düşürülebilir. Bu performans ölçüm aralığına göre değişir. Çözünürlük, ortalama, düzleştirme ve ölçüm noktalarının sayısı gibi ayarlar ölçüm amacına göre ayarlanabilir. Buna ek olarak tepe ve dipler tespit edilebilir, spektrum analiz edilebilir ve spektrum genişliği ve SMSR v.b bulunabilir. Ayrıca, WDM'deki spektrumda bulunan çoklu tepeler bölge işaretleyicileri koyarak analiz edilebilir. Opsiyonel dalgaboyu referansı kullanılarak analizör kalibrasyonu yapıldığında (özellikle WDM v.b sistemleri yüksek güvenilirlik ölçümlerini doğruluğunu gerçekleştirmede) kullanışlı olan dalgaboyu ayarı ±50 pm (1530 – 1570 nm)'dir. Module edilmiş spektrum ve darbe ışık kaynakları harici sinyal eşleşirime ve tepe tutan ölçüm fonksiyonları kullanılarak ölçülebilir. Ölçülen veri ve dalga şekli

MS9170B veri formatında, MS-DOS text formatında veya MS-Windows bitmap formatında diskete (FD) kaydedilebilir. Text ve Bitmap dosyaları popüler kelime işlemci ve tablo uygulamalarına kolaylıkla eklenebilir (Anritsu 2008).

### 3.3.2. Özellikler

MS9710B'nin özellikleri aşağıda listelenmiştir (Anritsu 2008).

- Hızlı ve doğru polarizasyon bağımlı ölçüm ve 1,55 μm bandta seviye doğrusallığı
- Geniş dinamik aralık (70 dB), yüksek hassasiyet (-90 dBm) ve optik zayıflatıcı kullanarak +20 dBm'e kadar ölçüm yapabilme (isteğe bağlı)
- Daha önceki analizörlerden daha küçük ve daha hafif
- Spektrum tepe algılama ve birçok cihazı değerlendirmek için dahili hesaplama fonksiyonlarıyla analiz
- Veri MS9710B formatında ve MS-DOS text ve MSWindows Bitmap formatında diskete kaydetme
- GPIB ve RS-232C arayüzleri ile harici kontrol
- Dalgaboyunun kalibrasyonu için dahili ışık kaynağı

### 3.3.3. Kullanıcı arayüzü

MS9710B ölçüm ayarları kart diye adlandırılan görüntülerden ekranlanır. Kartlar ekranın alt tarafındaki F1'den F7'ye kadar olan kart seçme tuşlarına basılarak seçilir. Her bir kart ekranın sağ tarafına ayarlanmış F1'den F7'ye kadar olan fonksiyon tuşları kullanılarak seçilen seçeneklere ilişkilendirilmiştir. Wavelength kartıyla başlar yedi kart önplan kartları ve Graph kartıyla başlayan yedi kart arkaplan kartları olarak adlandırılır. Background kartlarının görüntüleri F8 dönüştürme butonuna basılarak görüntülenebilir (Anritsu 2008).

### 3.3.3.a. Önplan kartları

Wavelength kartı: F1 – Dalgaboyunu ayarlar.

Level Scale kartı: F2 – Seviyeyi ayarlar.

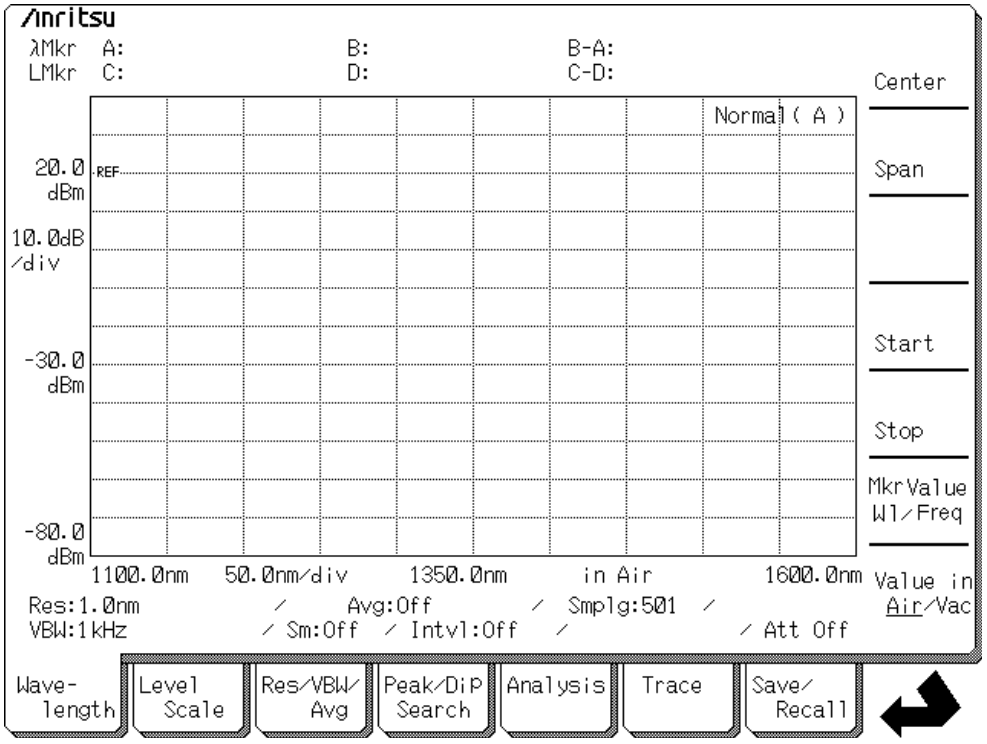
Res/VBW/Avg kartı: F3 – Çözünürlüğü, alınan ışık bant genişliğini ve ortalamasını ayarlar.

Peak/Dip/Search kartı: F4 – Tepe ve dipleri algılar

Analysis kartı: F5 – Dalga şeklini analiz eder.

Trace kartı: F6 – Hafıza ve işaretleri değiştirir.

Save/Recall kartı: F7 – Veriyi kaydeder ve geri çağırır.




**Şekil 3.16.** Önplan (Foreground) kartlarının görünüşü (Anritsu 2008).

### 3.3.3.b. Arkaplan kartı

Graph kartı: F1 – Grafik görüntüsünü dönüştürür.

Application kartı: F2 – Uygulama ölçüm fonksiyonlarını dönüştürür.

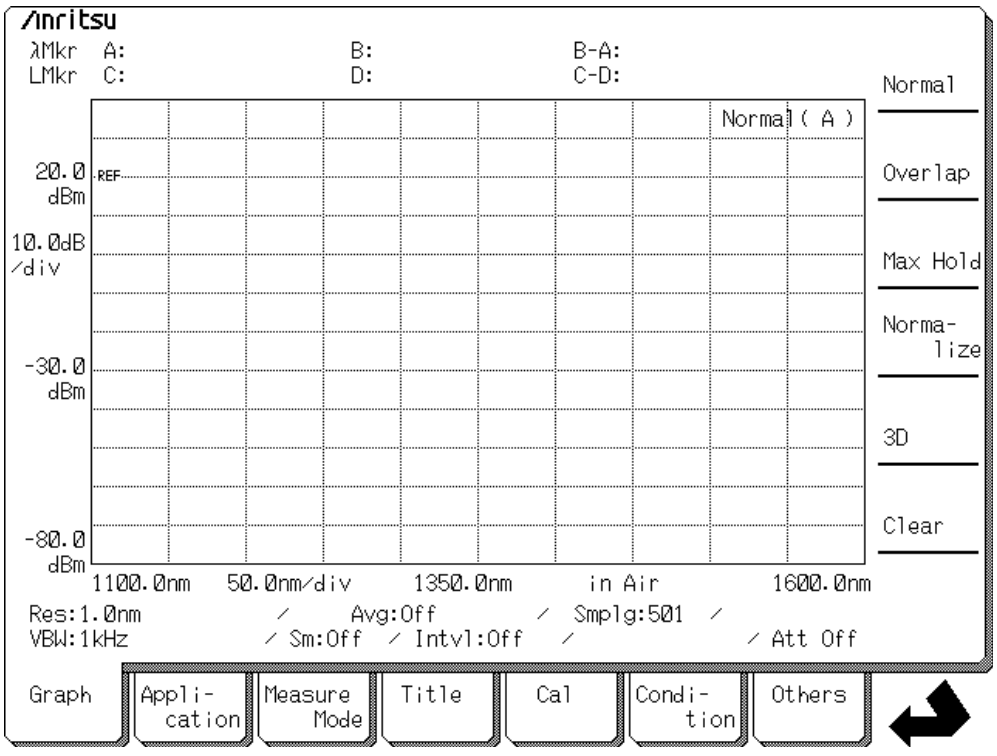
Measure mode kartı: F3 – Özel ölçüm modu ayarlar.

Title kartı: F4 – Ekran başlığını ayarlar.

Cal kartı: F5 – Kalibrasyon fonksiyonunu ayarlar.

Condition kartı: F6 – Ölçüm şartlarını kaydeder ve geri çağırır.

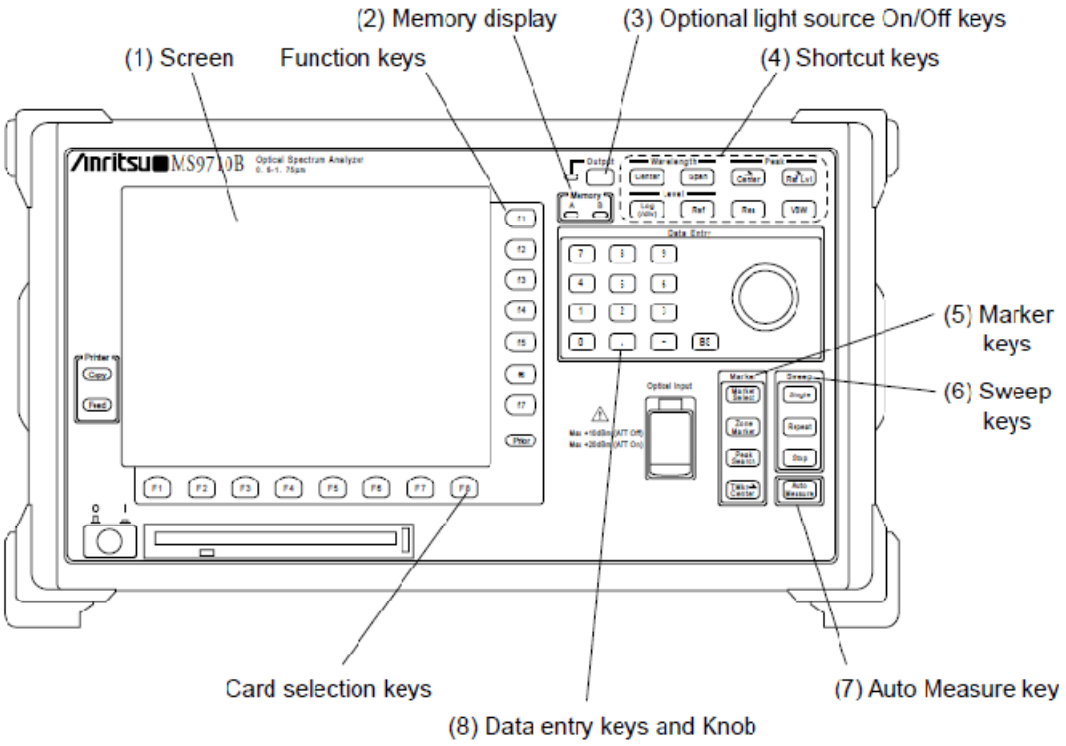
Others kartı: F7 – Diğer fonksiyonları ayarlar.




**Şekil 3.17.** Arkaplan (Background) kartlarının görünüşü (Anritsu 2008).

### 3.3.3.c. Ön panel

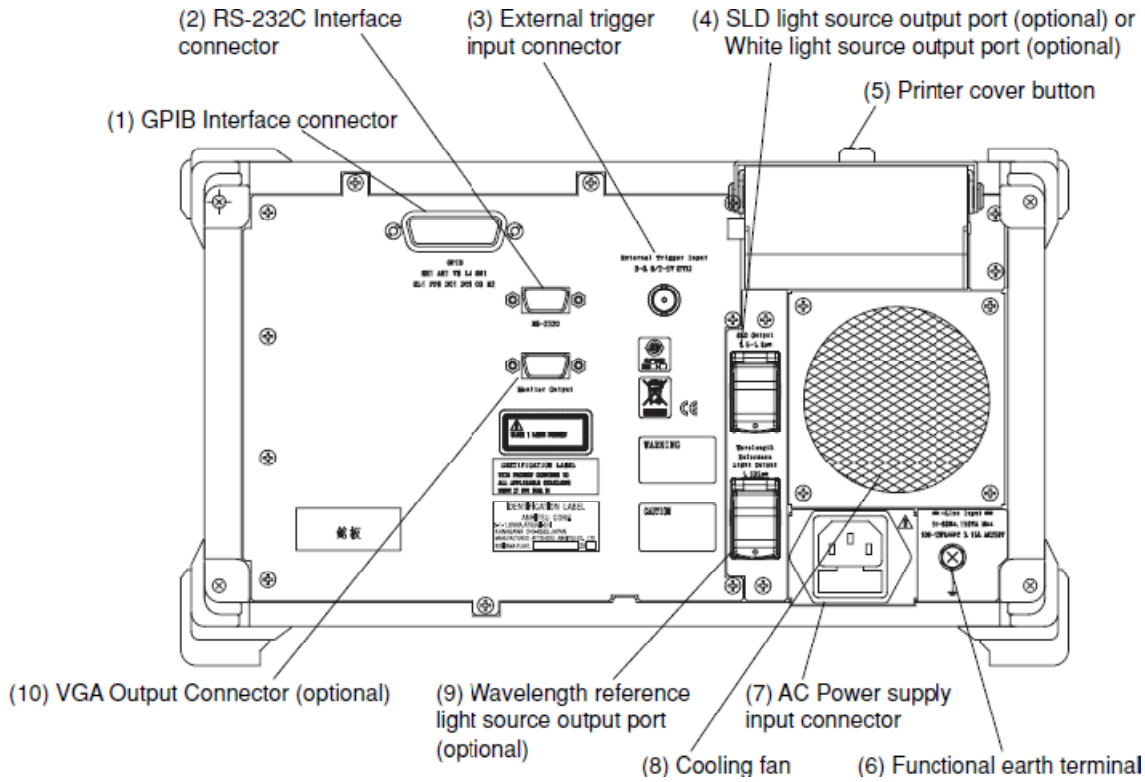
Bu kısımda MS9710B'nin ön paneldeki her bir kısmın fonksiyon ve isimleri tanımlanmıştır.




**Şekil 3.18.** MS9710B Ön panel görünüşü (Anritsu 2008)

**Screen (Ekran)** ölçülen spektrumu göstermesi yanı sıra seçilen analizör fonksiyonları için kartları gösterir. Ekranın alt tarafına hizalanmış F1'den F8'e kadar olan sekiz kart seçme tuşları kullanılarak görüntülenebilir. Her bir kart içindeki fonksiyon seçenekleri karşılık gelen yedi fonksiyon tuşuna ya da ekranın sağ kenarı boyunca hizalanmış F1'den F7'ye kadar olan fonksiyon tuşlarından birine basarak seçilebilir. Prior tuşu seçilen fonksiyonu iptal eder ve önceki şartlara döner. Prior tuşu aynı zamanda ayarları görüntüleme alanını silmek için de kullanılır.

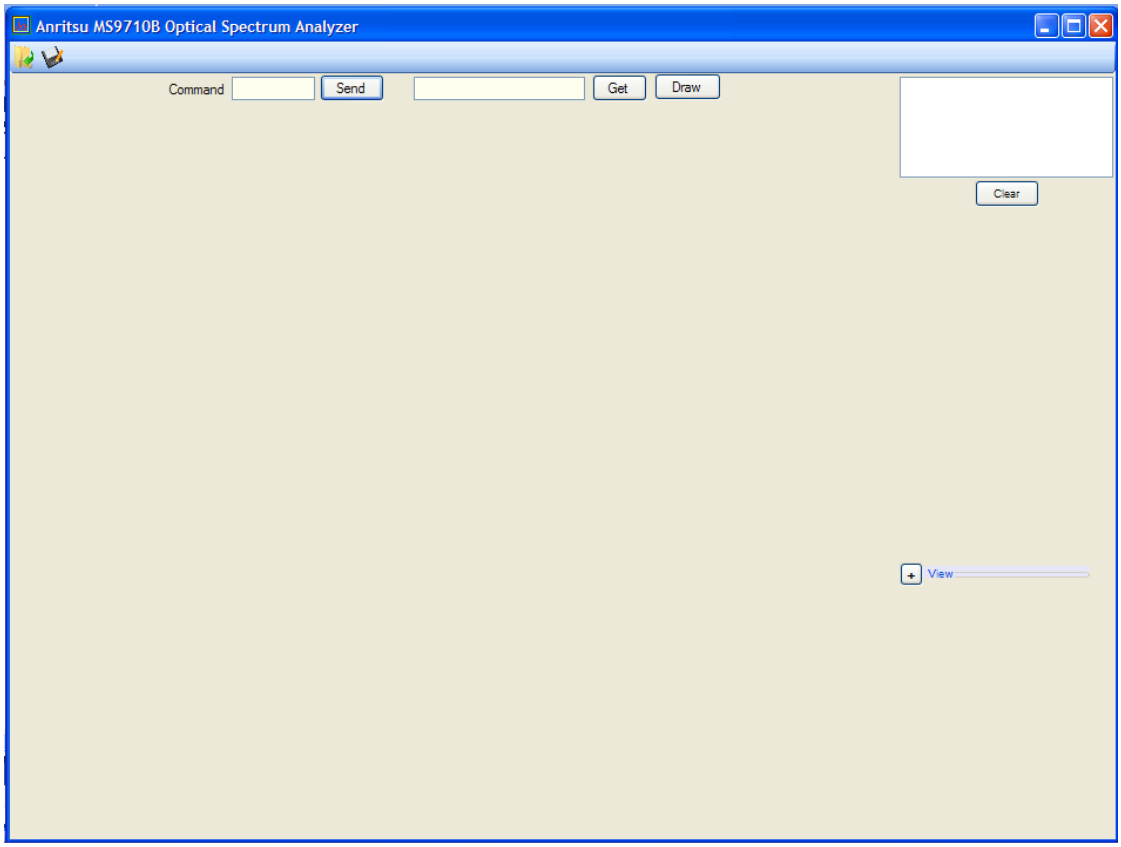
### 3.3.3.d. Arka panel



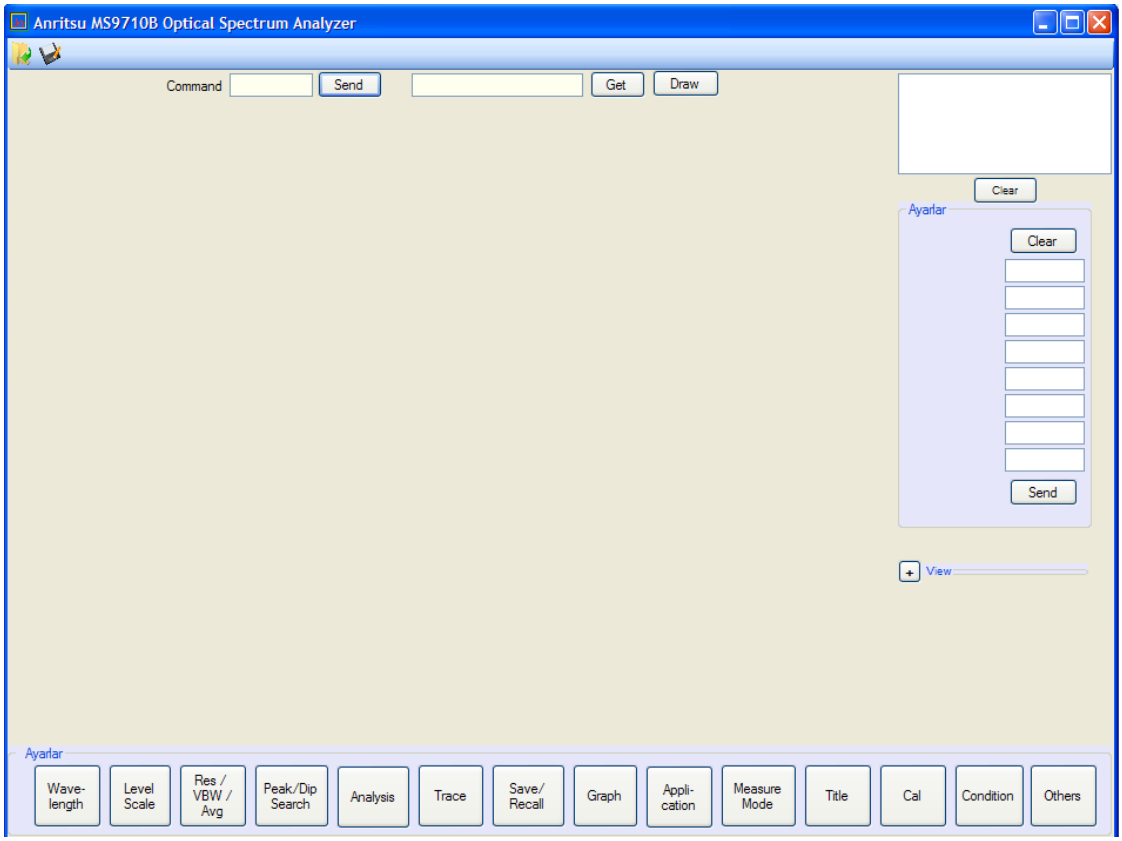

**Şekil 3.19.** Arka panel görünüşü (Anritsu 2008)

# 4. ARAŞTIRMA BULGULARI

Materyal ve yöntem bölümünde bahsi geçen MS9710B optik spektrum analizörünün uzaktan kontrolü için Visual C# 2010 görsel programlama dilinde bir kullanıcı arayüzü yapılmıştır. Bu arayüzde Visual C# Windows Forms uygulamaları kullanılmış olup bu forma gerekli kontroller eklenerek son haline getirilmiştir. Arayüzle cihaza GPIB standart komutları ve cihaz bağımlı mesajları gönderilebilir. Böylece bu arayüz sayesinde; cihaza uzaktan ölçüm yaptırılabilir, ölçüm sonuçları tekrar bu arayüzde grafik olarak çizdirilebilir, çizdirilen grafik üzerinde yakınlaştırma, uzaklaştırma gibi görünüm değişikliği yapılabilir, cihazdan alınan bilgilerle çizdirilen orijinal grafik ya da üzerinde değişiklik yapılmış bu grafiğin türevleri text veya bitmap olarak kaydedilebilir ve cihaz üzerinde bulunan fonksiyon tuşları form üzerinde çift tıklanarak buton şeklinde kullanılabilir. Oluşturulan bu kullanıcı arayüzü çalıştırıldığında karşımıza çıkan ilk ekran Şekil 4.1'de ve formun çift tıklanması sonucu fonksiyon tuşlarının görünür hale getirilmiş şekli ise Şekil 4.2'de gösterilmiştir.




**Şekil 4.1.** Oluşturulmuş kullanıcı arayüzünün açılış görünüşü




**Şekil 4.2.** Form çift tıklandığındaki ekran görüntüsü

Bu kullanıcı arayüzü oluşturulurken cihaz ile haberleşmek için genel amaçlı arayüz yolu (GPIB) komutları kullanılmıştır. Bu komutlar içerisinde cihazın başlangıç ayarlarına ayarlanması, zaman aşımının ayarlanması, bordun hazır olup olmadığının kontrolü gibi arayüz mesajlarıyla; cihazın istenilen ölçüm ayarlarının yapılması, ölçümün yapılması ve sonuçları kullanıcı arayüzüne gönderilmesi gibi cihaz bağımlı mesajlar kullanılmıştır. Kullanılan komutların bir kısmı Şekil 4.3, Şekil 4.4 ve Şekil 4.5'te gösterilmiştir.

```
private void Form1_Load(object sender, EventArgs e)
{
    if (CEC488.gpib_board_present() == true)
    {
        CEC488.boardselect(0);
        CEC488.initialize(21, 0);
        CEC488.settimeout(30000);
        CEC488.transmit("LISTEN 16,TALK 16", out status1);
    }
}
```

**Şekil 4.3.** Cihazın 16. adresten konuşma ve dinlemeye hazırlanması

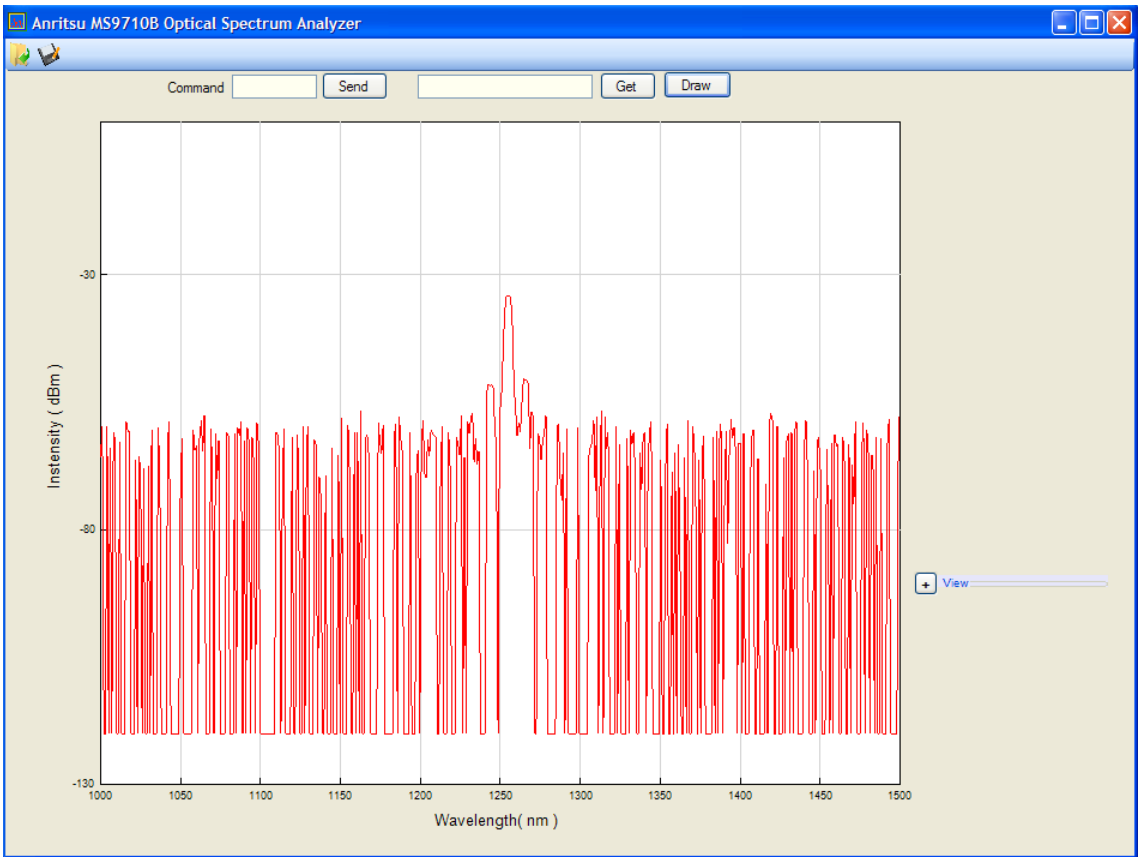
```
private void button2_Click(object sender, EventArgs e)
{
    string cevap, komut;
    komut = textBox1.Text;
    if (komut != "")
    {
        CEC488.send(adres, komut, out status1);
        CEC488.enter(out cevap, 65000, out uzunluk, adres, out status2);
        if (status2 == 0)
            textBox2.Text = cevap;
        textBox1.Text = "";
    }
    else
        MessageBox.Show("Cevap alınamadı");

}
```

**Şekil 4.4.** Textbox1'e yazılan komutun cihaza gönderilmesi

```
CEC488.transmit("MTA LISTEN 16 DATA 'DQA?' END MLA TALK 16", out status1);
```

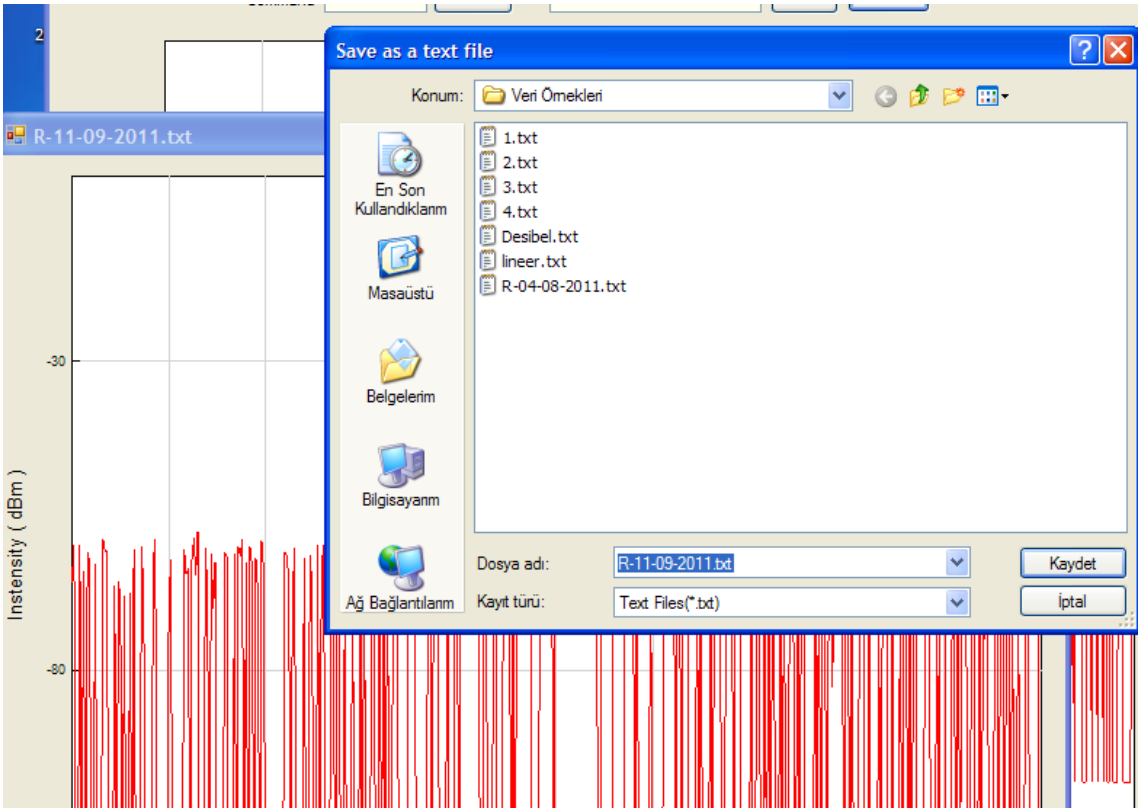
**Şekil 4.5.** Cihazın ölçüm değerlerinin 16. GPIB adresinden sorgulanması

Şekil 4.6'da cihaza gönderilen komut ile ölçüm yaptırıldıktan sonra cihaz ekranındaki spektrumun arayüze aktarılmış grafiksel şekli görünmektedir.

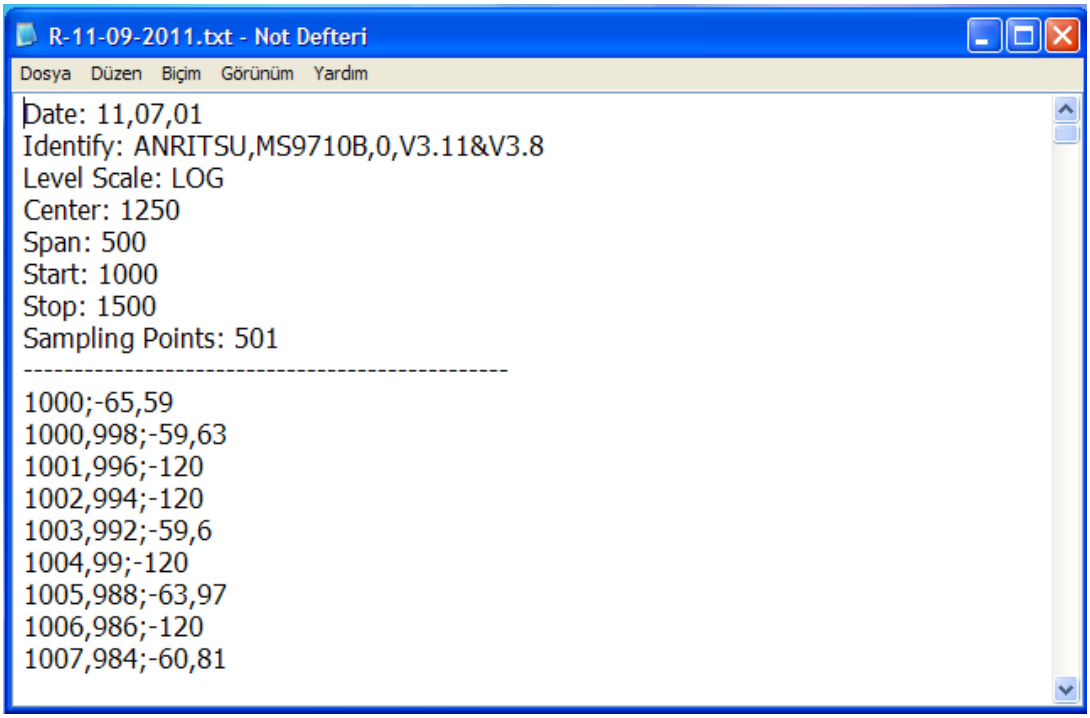



**Şekil 4.6.** Draw butonuna basılarak cihazdaki spektrumun arayüze çizdirilmesi

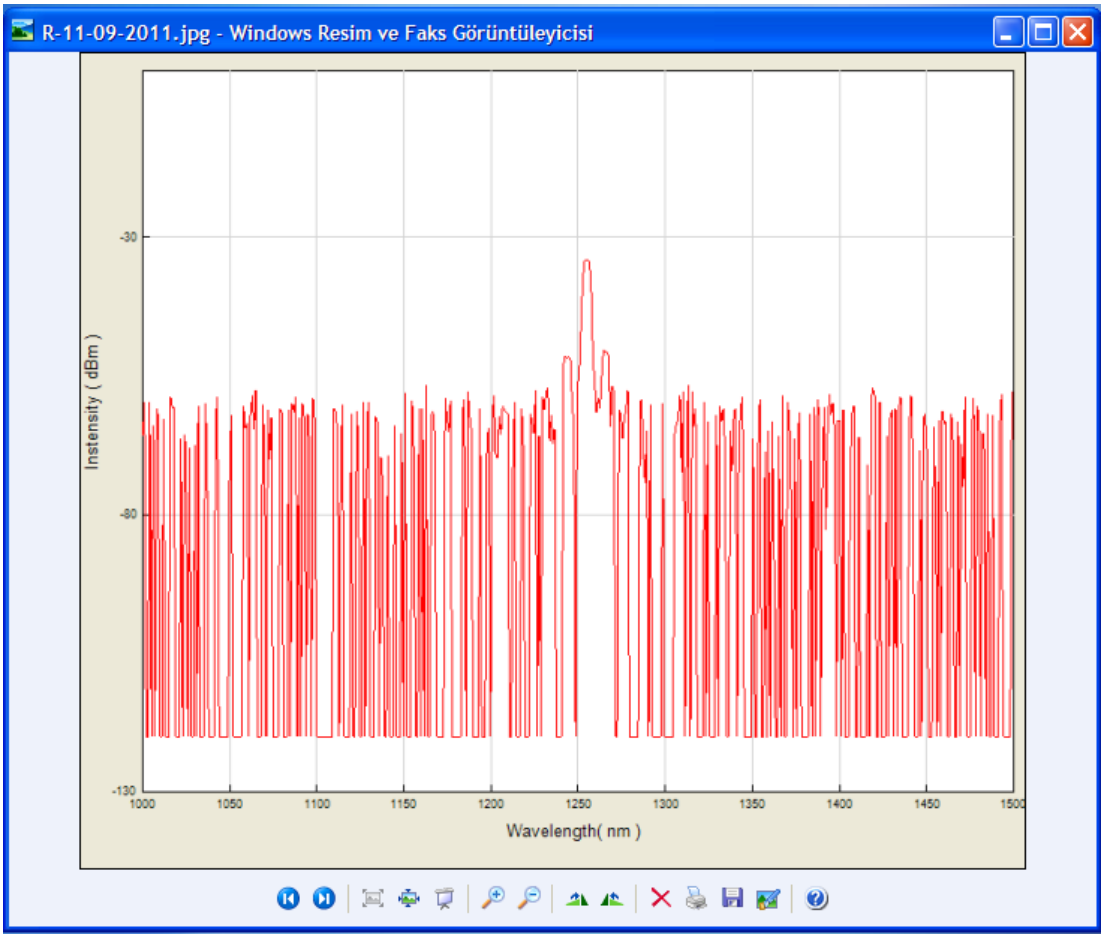
Arayüz ekranına cihazdan alınan spektrum çizdirildikten sonra istenilirse bu grafik text veya bitmap olarak bilgisayara kaydedilebilir. Şekil 4.7, Şekil 4.8 ve 4.9'da kaydet butonuna basıldıktan sonra gelen ekran ve daha sonra text ve bitmap olarak kaydedilen dosyaların ekran görüntüleri verilmiştir.




**Şekil 4.7.** Save butonuna basılarak kaydetme işleminin yapılması




**Şekil 4.8.** Text olarak kaydedilen dosyanın içeriği




**Şekil 4.9.** Bitmap olarak kaydedilen grafik

# 5. TARTIŞMA ve SONUÇ

Elektronikte teorik olarak yapılan çalışmaların daha sonraki teorik çalışmalara ışık tutması için uygulamalarla desteklenmesi gerekir. Birçok fakülte ve enstitüde eski haberleşme protokollerini barındıran ölçme ve test cihazları hala kullanılmaktadır.

Yapılan çalışmada kullanılmış olan MS9710B optik spektrum analizöründe, ölçüm sonuçları yalnızca diskete (floppy diske) kaydedilebilmektedir. Bilindiği üzere günümüzdeki masaüstü bilgisayar, laptop ve netbook gibi bilgisayarlarda floppy disk bulunmamaktadır. MS9710B, bilimsel çalışmalarda ihtiyaç duyulan ölçümleri gerçekleştirebilmesine rağmen sonuçların kaydedilip daha sonra erişilebilmesi için bilgisayara iletiminde sadece RS-232 ve GPIB haberleşmelerini sunmaktadır. Çalışmada cihazın uzaktan kontrolü için GPIB kullanılmasının sebebi birden çok test veya ölçüm cihazının bu sisteme dahil edilebilmesidir.

MS9710B optik spektrum analizörünün uzaktan kontrolü için Visual C# programında bir kullanıcı arayüzü yapılmış olup Keithley firmasının USB-GPIB kartı kullanılarak cihaz ve PC arasında iletişim sağlanmıştır. Daha sonra yapılan program aracılığıyla cihaza ölçme ve ölçüm sonuçlarını gönderme komutları verilerek cihazdan alınan veriler doğrultusunda arayüz ekranına cihaz ekranındaki spektrumun aynısı çizdirilmiştir. Çizilen grafik üzerinde başlangıç, bitiş dalgaboyu gibi özellikler değiştirilerek oluşturulan grafik senkronize olarak her iki ekranda görüntülenmiştir. Orijinal ya da üzerinde değişiklik yapılmış grafik her an kaydet butonu ile kaydedilebildiği için değişik zamanlarda text ve bitmap formatında bilgisayara kaydedilmiştir. Cihaza bağlı olunmasa dahi daha önce kaydedilmiş ölçüm sonuçları kullanıcı arayüzüne taşınabilir.

Sonuç olarak aslında ölçme ve testte yeterli olan bir cihazın yapılan çalışmayla atıl duruma düşmesi engellenerek mevcut teknolojiye ayak uydurmasına olanak sağlanmıştır.

**KAYNAKLAR**

Anritsu Corporation, 2007. MS9710B Optical Spectrum Analyzer Remote Control, 268p, Anritsu, Japan.

Anritsu Corporation, 2008. MS9710B Optical Spectrum Analyzer Operation Manual, 264p, Anritsu, Japan.

Axelson, J., 2005. USB Complete:Everything You Need to Develop Custom USB Peripherals, Labview Research, 572 p, Madison, U.S.A..

Freeman, A., 2010. Introducing Visual C# 2010, 1321p, U.S.A..

Keithley Instruments Inc., 2010. Models KPCI-488LPA and KUSB-488B Reference Manual. Keithley, 74p, Cleveland, Ohio, U.S.A..

Microchip Technology Inc., 2004, DS39632B, PIC18F2455/2550/4455/4550 Data Sheet, Microchip Technology Incorporation.

Park, J. and Mackay, S., 2003. Practical Data Acquisition for Instrumentation and Control Systems. Newnes, 425p, Burlington.

QISI (Quality Instrumentation Solutions Inc.), 2007. GPIB-488 Programming Reference Manual for the CEC API., QISI, 134p, Norton, U.S.A..

Şahin, C., Dayanık, A. ve Altınbaşak, C., 2006, PIC Programlama Teknikleri ve PIC16F877A, 1. baskı, Altaş Yayınları, 526 s, İstanbul.

Tan, W.M., 1997, Developing USB PC Peripherals Using The Intel 8x930Ax USB Microcontroller, Annabooks, 189 p, San Diego, U.S.A..

Hewlett Packard Co., Intel Co., Microsoft Co., NC., ST-Ericsson Co. and Texas Instruments, 2011. Universal Serial Bus 3.0 Specification, http://www.usb.org/developers/docs (16.09.2011).

Xu, J., 2007. Practical C# Charts and Graphics. UniCAD, 571p, Phoenix, U.S.A..

## ÖZGEÇMİŞ

1986 yılında Erzurum'da doğdu. İlk ve ortaöğretimini bu şehirde tamamladıktan sonra 2003 yılında Kahramanmaraş Sütçü İmam Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği Bölümünü kazandı. Lisans öğreniminden sonra, 2009 yılında Gümüşhane Üniversitesi Kelkit Aydın Doğan Meslek Yüksekokulu Elektronik ve Otomasyon Bölümünde öğretim elemanı olarak göreve başladı. 2009 yılında Atatürk Üniversitesi Fen Bilimleri Enstitüsü Elektrik ve Elektronik Mühendisliği Anabilim Dalı'nda yüksek lisans öğrenimine başladı.

Gümüşhane Üniversitesi Kelkit Aydın Doğan Meslek Yüksekokulu'nda 2010 yılından beri Müdür Yardımcısı olarak görevine devam etmektedir. Evli ve İngilizce bilmektedir.