ADIM ADIM

USB VE UYGULAMALAR

USB DEVELOPER GROUP

```
bool bDeviceFound = false;

hDeviceSet = Win32.SetupDiGetClassDevs(ref HidGuid,
                                       0,
                                       IntPtr.Zero,
                                       Win32.DIGCF_PRESENT |
                                       Win32.DIGCF_DEFIVEINTERFACE
                                       );

if (hDeviceSet == Win32.INVALID_HANDLE_VALUE) return false;

Win32.SP_DEVICE_INTERFACE_DATA devInterfaceData = new Win32.SP_DEVICE_INTERFACE_DATA();
Win32.SP_DEVICE_INTERFACE_DETAIL_DATA devInterfaceDetailData = new Win32.SP_DEVICE_INTERFACE_DETAIL_DATA();

while (Win32.SetupDiEnumDeviceInterfaces(
                        hDeviceSet,
                        IntPtr.Zero,
                        ref HidGuid,
                        iDeviceCount,
                        devInterfaceData
                        ))
{

    //Adım 1
    Success = Win32.SetupDiGetDeviceInterfaceDetail(
                                hDeviceSet,
                                devInterfaceData,
                                IntPtr.Zero,
                                0,
                                ref iSize,
                                IntPtr.Zero
                                );

    IntPtr DetailDataBuffer = Marshal.AllocHGlobal(iSize);
```

USB Arabirimini kullanarak proje geliştirebilmeniz için gerekli herşey....

Yazan: Ahmet ATAR

www.usbdevgroup.blogspot.com

İÇİNDEKİLER

Bölüm 1

Bölüm 2

Bölüm 3

Bölüm 4

Bölüm 5

Adım Adım USB Cihaz Tasarımı
- Yonga Kodu'nun Yazılması
- C18 USBHid Firmware'nin incelenmesi
- Firmware'ın Test Kartına Yüklenmesi ve İlk Deneme
- Listelemenin Device Monitoring Studio İle İzlenmesi
- Listeleme Sonrası Registry

Bölüm 6

Windows Programı'nın Tasarımı ve Detaylar
- API Nedir?
- Sürücü nedir ve nasıl yazılır?
- Sürücülerin C# İle Yazılmış Yardımcı Araçla İncelenmesi
- INF Dosyası ve İçeriği
- USBHid Windows Uygulaması
- USB Haberleşmesinde ve Diğer İşlemlerde Kullanılacak API'ler
- Win32 Mesaj Sistemi ve Cihaz Takibi
- C# ile USB Windows Uygulamasının Geliştirilmesi
- C# ile Hazırlanmış USBApplication Programının İncelenmesi
- USBManagement.dll
- Ana Uygulama

Bölüm 7

Uygulama Programının ve Cihaz'ın Beraber Kullanılması
Cihaz'a Veri Göndermek Ve Okumak
Veri Trafiğinin Device Monitoring Studio İle İzlenmesi
C# İle Geliştirilmiş Diğer Yardımcı Uygulamalar

Bölüm 8

RS232 İle Tasarlanmış Cihazların USB'ye Yükseltilmesi
- FT232BM Yongası ve Özellikleri
- Sürücü Desteği
- Donanım Tasarımı
- Örnek Firmware'in İncelenmesi
- Windows Uygulamasının Geliştirilmesi

Bölüm 9

USB Uygulamaları
- USB-LCD Uygulaması
- USBtoRS232
- DS1621 ile Sıcaklık Ölçümü
- HT90S32 ile CALLER ID Uygulaması
- ATX-34 ve ARX-34 ile Alıcı/Verici Uygulaması
- Internet Üzerinden Ev Otomasyonu
- USB Motor Kontrol

Bölüm 10

Sinyaller ve Şifreleme
Güç Seçenekleri ve Kablolama

Giriş

Günümüzde birçok elektronik cihazın USB arabirimini içerdiğini görmekteyiz.Hiç yanımızdan ayırmadığımız MP3 çalarımız, bilgilerimizi cebimizde taşımamıza olanak sağlayan usb disklerimiz buna en güzel örnektir.Bu sebepten dolayı bizimde tasarladığımız elektronik ürünlerin eğer bilgisayar ile iletişim kurması gerekiyorsa arabirim olarak USB'yi desteklemesi kaçınılmaz olmaktadır.Çünkü günümüzde seri, paralel gibi birçok standart yerini USB'ye bırakmış durumda.Eskiden paralel veya seri porttan çalışan yazıcılarımız bile bugün USB arabirimi ile tasarlanmaktadır.
Bu yüzden bu makalenin hazırlanma amacı elektronikle gerek hobi amaçlı, gerekse profosyonel olarak ilgilenen kişilere "hazır yazılım ve donanım" kullanmadan direkt kendi USB arabirimli cihazlarını en basit yoldan tasarlamaları için hazırlanmıştır.Birçok elektronik içerikli sitede USB arabirimli cihaz tasarlamak isteyen kişiler bu gereksinimlerini ya hazır bir HEX dosyası ve donanım bulup gidermekte ya da çaresiz diğer eskiyen protokolleri kullanmakla yetinmektedir.İşte bu makalenin hazırlanış amacı elektronikçileri "hazırcı zihniyet" anlayışına iten bu etkenleri ortadan kaldırmak ve Türkiye'de bu konuda neredeyse yok denecek kadar az bulunan Türkçe döküman eksikliğini gidermektir.
Makale toplam on bölümden oluşmaktadır.Birinci bölümde USB'nin gelişim tarihi ve avantajlarına çok kısa bir şekilde giriş yapılmıştır.İkinci bölümde ise tasarım esnasında kullanılacak devre ve programlar tanıtılmaktadır.Üçüncü bölümde USB Protokolünün detayları ve elemanları kafa karıştırmayacak şekilde incelenmektedir.Dördüncü bölümde Windows'un bünyesinde bulunan HID sınıfı incelenmektedir.Çünkü ilerki bölümlerde tasarlanacak cihaz bu sınıfa uygun olacaktır.Beşinci bölümde teorik bilgiler edinildikten sonra adım adım USB cihaz tasarımına geçilmektedir.Bu bölümde tasarım bitimi listeleme süreci ve cihazın Windows'a tanıtılmasıda konu alınmıştır.Altıncı bölümde tasarlanan cihaz ile heberleşecek uygulama yazılımı tasarlanmaya başlanmış bu konu için daha önce hazırlanmış olan USBApplication programı incelenmiştir.Bu bölümde aynı zamanda sürücü nedir ve transferin neresinde bulunur, Win32 altyapısı gibi konulara da değinilmiştir.Yedinci bölümde donanım ve yazılım tasarımı bittikten sonra test uygulamasına geçilmiş ve tasarım ilk kez test edilmiştir.Sekizinci bölümde daha önce seri protokol ile büyük tasarımlar yapmış elektronikçilerin, sadece donanımlarında ve yazılımlarında yapacakları ufak değişikliklerle ürünlerini nasıl USB arabirimine yükseltecekleri incelenmiştir.Dokuzuncu bölümde ise, USB protokolü kullanılarak geliştirilmiş toplam yedi adet uygulama, firmware ve bilgisayar yazılım açıklamaları ile yer almaktadır.
Son bölüm olan onuncu bölümde ise USB güç idaresi ve kablolama hakkında bilgiler verilmiş, aynı zamanda sinyaller ve şifreleme konularınada değinilmiştir.
Bu makale hazırlanırken gerek uygulamalarda, gerekse teorik konularda, okuyanların C# ve C konularına yeteri düzeyde hakim oldukları varsayılmıştır.Çünkü tüm bu konular Microchip firmasının 18xxx serisi denetleyicileri için ürettiği C18 derleyicisi kullanılarak C ile yazılmış, bilgisayar uygulamaları ise DOTNET dillerinden C# ile hazırlanmıştır.Gerekli araç ve gereçlerin konu anlatımı boyunca hangi dizinlerde yer aldığı sırası ile açıklanmıştır.

Tüm elektronikçilere faydalı olması dileği ile....
Ahmet ATAR

Bölüm 1

USB Tarihçesi

USB arabiriminin geliştirilmesi Hewlett Packard firması ile başlamıştır.Fakat o zamanlar HP Arabirim Bus'ı olarak anılmaktaydı.Bu arabirimin sadece HP'ye yani tek bir firmaya ait olması Lisans ücretleri ve buna benzer sebeplerden dolayı istenmeyen bir durum olduğundan USB standartlarını belirlemek üzere bir örgüt kurulmuştur. Bu örgütte bulunan şirketler Intel,Microsoft,NEC, Philips HP ve Compaq'dır.Ocak 1996 tarihinde ilk olarak USB 1.0 sürümü devreye girmiş,hemen ardından Eylül 1998' de ise USB 1.1 sürümleri hazırlanmıştır.
USB 1.1 ile gelen özelliklerden bir tanesi Kesme OUT transfer tipi'nin ve yüksek hız desteğinin eklenmesidir. (Bu transfer tipi ve diğerleri ilerki ünitelerde incelenecektir.)Aralık 2000'de ise mini-B tipi konnektör eklenmiştir.
USB arabirimine destek veren ilk işletim sistemi 98 kadar kuvvetli olmada Windows 95'dir.Fakat bu arabirim 98 işletim sistemi ile yaygınlaşmıştır.
Tüm gelişmelerden sonra en büyük adım olan USB 2.0 geliştirildi.20 kat olması hedeflenen transfer hızı 40 kat olarak belirlenmiş ve transfer hızı saniyede 480MBit yani 60MB olmuştur.

Avantajları ve Dezavantajları

USB arabiriminin avantajları denildiğinde akla ilk gelen yüksek işlem hızı ve hata denetiminin son derece güvenli olmasıdır.Böylece tasarımcıları yazılım ve donanımlarında hata denetimi ile uğraşmaktan kurtarmaktadır.USB arabirimi sinyallere fonksiyonlar yüklemediğinden birkaç USB portuna takılı cihaz aynı hat üzerinden farklı işlevleri yerine getirebilir.Paralel porta takılı bir cihaz düşünün.Bu cihaz paralel portun bir veya birkaç bitini kendisi için tahsis edip özel bir amaçla kullanıyorsa bu diğer cihazlar için bir engel oluşturacaktır.
Bir diğer avantaj ise güç sarfiyatının düşük olmasıdır.Tasarımlar sayesinde kullanılmayan cihazların güçleri kesilir fakat gerektiğinde tekrar haberleşmeye hazırdırlar.
USB tasarımında kullanılacak devre elemanları pahalı değildir.Örneğin bu makaledeki projelerimizde kullanacağımız PIC18F4550 denetleyicisi Microchip firması tarafından üretilmiş, ek özellikleri ile birlikte USB modülünüde bünyesinde barındırmaktadır ve fiyatı yaklaşık 15YTL kadardır.Bu yonganın SPI,I/O,CCP,USART,ADC,TIMERS ve kesmeler gibi ek özellikleri sayesinde hem USB haberleşmesi halledilir hemde tasarımla ilgili ek işlemler tek bir yonga ile yapılabilir.
USB'nin dezavantajları denildiğinde ise akla ilk gelen eski donanımlarla uyumsuz olmasıdır.USB desteği olmayan bir sisteme USB cihazını bağlamanın tek yolu dönüştürü kullanmaktır.Fakat buda USB işlevselliğini gölgelemektedir.
USB arabirimi masaüstü arabirimi olarak tasarlandığından mesafe sınırlamaları vardır.Kablo boyutu düşük hızlı cihazlar için 3 metre tam hızlı ve yüksek hızlı cihazlar için 5 metre olabilir.Fakat hub kullanılarak bu sınır azda olsa aşılabilir.
Cihazlar kendi aralarında haberleşemezler.İki mikrodenetleyici yonga seri protokol kullanarak örneğin 9600baud hızında haberleşebilirken USB sistemlerde bu söz konusu değildir.Haberleşmeyi sadece PC yönetir, başlatır ve sonlandırır.
USB'nin dezavantajları hakkında bir örnek daha verirsek buda protokolün çok karmaşık olmasıdır.Aynı zamanda USB cihazların satışlarında, cihazın üretici kimliğine sahip olması gerekir.Bu USB-IF'den 1500$ karşılığında alınabilir.Yani yüksek çaplı projeler tasarlanmak isteniyorsa USB'nin çokda ucuz olduğu söylenemez.
Tüm bu dezavantajların yanında USB arabirimin diğer avatajları göz önüne alındığında başka protokollerle çalışmak zaman kaybından başka birşey değildir.

Bölüm 2

USB Projesi Geliştirmek İçin Gerekli Elemanlar

USB Denetleyici Seçimi

Piyasada USB projelerinde kullanılmak üzere farklı firmalarca tasarlanmış birçok USB denetleyicisi mevcuttur. Bazıları bir mikrodenetleyiciye arabirim olarak bağlanabilirken, bazıları da direkt denetleyici içerisinde yer almaktadır.
Bir denetleyiciye arabirim olarak bağlanabilen USB arabirim yongaları maliyet, eleman fazlalığı ve giriş/çıkış uçlarının bazılarının kendilerine tahsis edilmeleri gibi unsurlardan dolayı çok fazla tercih edilmemelidir.Fakat bu yapılacak tasarıma ve USB arabirim yongasına göre değişmektedir.Örnek vermek gerekirse USBN9604 gibi bir USB arabirim yongası, bir denetleyiciye hem paralel hemde seri olarak bağlanabilir.Aynı zamanda piyasa fiyatı kendi kategorisinde bulunan diğer yongalara göre daha ucuzdur.Fakat yinede bunun haricinde birde denetleyici yonga fiyatı olduğundan On-Chip USB yongalar daha ucuza gelmektedir.
Bu sorunların yanında tasarım yapılacak USB yonganın kolay bulunabilmesi de belli başlı bir olaydır.Aynı zamanda kullanılacak yonganın iç mimarisine hakim olmak, internetten hazır kod ve kaynak bulmak da son derece önemli etkenlerdendir.
İşte bu sorunları göz önünde bulundurarak projelerimizde, tek chip üzerinde USB arabirim modülü ve diğer çevresel donanımları bulunan PIC18F4550 yongasını kullanacağız.Bu chip'e alternatif olarak giriş/çıkış sayısı daha az olan PIC18F2550'de kullanılabilir.

PIC18F4550'nin USB Özellikleri

PIC18F4550'nin dahili USB birimi bulunmaktadır ve temel özellikleri aşağıda listelenmiştir;

* USB 2.0 uyumludur.
* Low Speed(1.5Mb/s) ve Full Speed(12Mb/s) hızlarını desteklemektedir.
* Kontrol, Kesme, İzokron ve Yığın transferleri desteklemektedir.
* 32 Adet Uçnoktası vardır.(Çift yönlü 16 adet)
* 1KB USB Ram belleği(Dual Access)
* Dahili voltaj regülatörü
* Dahili Pull-Up dirençleri

PIC18F4550'nin USB işlemleri için ayırdığı toplam 22 adet register'ı mevcuttur.Bunların en önemli olanları ve projelerimizde sıkça kullanacaklarımız aşağıda incelenmiştir.Daha fazla bilgi ve diğer bitlerin özellikleri için chip'in datasheet'i incelenebilir.
Her USB destekli microdenetleyicide olduğu gibi PIC18F4550'ninde SIE motoru bulunur.(Serial interface Engine)SIE gönderilen paketlerin çözülmesinde ve ilgili register'ların set edilmesinde aynı zamanda gönderilecek verilerinde USB paket formatına dönüştürülmesinde büyük bir rol üstlenir.PIC18F4550'nin SIE motoru dahili tranceiver'dan faydalanabileceği gibi dış bir tranceiver'ada bağlanabilir.Dahili bir 3.3 volt regülatörü vardır ve 5 volt'luk uygulamalarda dahili tranceiver'a güç sağlar.Hem SIE'nin hemde CPU'nun USB RAM'ine direkt hafıza erişimi mevcuttur.

USB Control Register – UCON --

U-0	R/W-0	R-x	R/C-0	R/W-0	R/W-0	R/W-0	U-0
—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—
bit 7							bit 0

Şekil-1) USB Kontrol Yazmacı

PPBRST bitinin set edilmesi tüm Ping-Pong Buffer işaretçilerinin düzenlenen tamponlara yerleştirilmesini sağlar.0 olarak bırakılırsa Ping-Pong buffer'lar görmezden gelinir. Ping-Pong buffer özelliği birçok denetleyicide bulunur.
Bu özellik bilgi setinin tampona yazılmasının hemen ardından ikinci setinde tampona yazılmasını sağlar.Böylece ilk veri gönderilirken ikinci veri seti'de gönderime hazır olur.Bu alım işleminde de aynıdır.

SE0 biti USB bus'ında Single Ended Zero durumu oluştuğunda set olur.Bu durum D+ ve D- hatlarının aynı anda yüksek olduğu özel bir durumdur.

PKTDIS biti SIE'nin paket aktarmasını ya da almasını aktif yapmak üzere kullanılan bir bayraktır.Bu bit bir SETUP paketi alınmasının ardından SIE tarafından set edilir ve paket aktarma/alma işlemi pasif yapılmış olunur.SIE'nin paket alma/aktarma işlemine tekrar izin vermek için bu bit CPU tarafından sadece temizlenebilir.

USBEN biti USB modülünü aktif/pasif yapar.Eğer cihaz bus'a bağlı ise ve bu bit set edilirse cihaz attached durumuna geçer, temizlenirse cihaz bus'dan çıkatılmış gibi olur.(Detached)Böylece cihaz'ın bus ile olan ilişkisi yazılımsal olarak kontrol edilebilir.Aynı zamanda bu bit set edildiğinde tüm PPBI yazmaçlarını 0'a kurar, USB regülatörünü dahili pull-up dirençlerini aktif yapar(Eğer ayarlandıysa)

RESUME biti PC'ye resume sinyali göndermek için kullanılır.Resume sinyali göndermek için bu bit yaklaşık 1 ile 13ms arası set edilmeli ardından temizlenmelidir.Askı durumundaki bir cihaz PC'ye haberleşmeyi canlandırması için Resume sinyali gönderebilir.

SUSPND biti USB modülün suspend duruma yani askıya geçmesini sağlar.Bu bit set edilerek SIE'nin clock kaynağı pasif yapılmış olunur.Bus'da 3ms işlem görmeyen bir cihaz suspend durumuna geçmelidir.Bu durum IDLEIF bayrağının set olmasıyla anlaşılabilir.Bu durumda SUSPND biti set edilerek suspend moduna geçilmelidir.Bus'da takrardan faliyet başlaması durumunda ACTVIF biti set olur.Bu durumda SUSPND biti temizlenmeli ve SIE'nin clock kaynağı aktif edilmelidir.

USB Configuration Register – UCFG --

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
UTEYE	UOEMON(1)	—	UPUEN(2,3)	UTRDIS(2)	FSEN(2)	PPB1	PPB0
bit 7							bit 0

Şekil-2) USB Konfigrasyon Yazmacı

UTEYE biti düşük hızlı cihazlarda J-K-J-K, tam hızlı cihazlarda ise K-J-K-J sinyelleri test'i için kullanılır.Daha fazla bilgi için datasheet'e bakın.

UPUEN biti dahili pull-up'ları aktif etmek için kullanılır.Eğer FSEN biti set edilmişse D+ pini, edilmemişse D- pini yükseğe çekilir.Böylece dahili pull-up'lar sayesinde dışarıdan full-speed veya low-speed için D+ ve D- pinlerini direnç ile yükseğe çekmeye gerek kalmaz.Bu bit set edildiğinde D+ ve D- pinlerinden herhangi biri dahili regülatör sayesinde direkt 3.3V'a çekilir.

FSEN biti tam hız ya da düşük hız seçimde kullanılır.PIC18f4550 sadece bu hızları desteklemektedir.Bu bit set edilirse harici saat girişi 48MHz, set edilmezse 6MHz olmalıdır.

PPB0-PPB1 bitleri Ping-Pong Buffer konfigrasyonu için kullanılır.Bu bitlere daha sonra değineceğiz.

USB Status Register –USTAT--

U-0	R-x	R-x	R-x	R-x	R-x	R-x	U-0
—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI(1)	—
bit 7							bit 0

Şekil-3) USB Status Yazmacı

ENDP0-ENDP3 arası bitler isteklerin ya da verilerin hangi uçnoktaya geldiğini gösterir.PIC18f4550'nin toplam 16 adet uçnoktası olduğundan bu dört bit ile uçnokta adresi belirlenmiş olur.

DIR biti transferin yönünü belirtir.Bu bit set ise transfer cihazdan PC'ye doğru(IN transfer) aksi halde PC'den cihaza doğrudur.(SETUP veya OUT transfer)

PPBI biti son işlemden sonra Ping-Pong Buffer işaretçisi'nin Odd/Even durumunu gösterir.

USTAT transfer hakkında bilgiler içerdiğinden önemlidir.Bir işlem tamamlanıp TRNIF biti set edildikten sonra USTAT SIE tarafından otomatik olarak güncellenir.USTAT SIE tarafından korunan 4 byte'lık bir FIFO'ya sahiptir.Bu FIFO dolu iken yeni bir istek gelmesi halinde SIE otomatik olarak NAK ile yanıt verecektir.Bir işlem tamamlandıktan sonra USTAT SIE tarafından otomatik olarak update edilir.

USB Endpoint n Control Register – UEPn --

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL(1)
bit 7							bit 0

Şekil-4) USB Uçnokta n Yazmacı

EPHSHK biti set edilirse uçnokta elsıkışma'sı aktif hale gelir.Bu bit set edilmediği takdirde elsıkışma olmaz.Bu daha çok izokron transferlerde kullanılır.Çünkü izokron transferlerde verinin koşulsuz alındığı kabul edilir ve veri akışı süreklidir.Fakat diğer transfer tiplerinde bu bit set edilerek elsıkışma aktif edilmedir.

EPCONDIS biti uçnokta'nın SETUP,IN veya OUT için kullanılmasını kontrol eder.Eğer EPOUTEN ve EPINEN bitleri set edilmişse, bu bitin set edilmesi halinde uçnokta sadece IN ve OUT işlemleri için kullanılır.Eğer set edilmezse SETUP işlemleri için, yani kontrol transferler için kullanılır.Fakat IN ve OUT işlemlerine de izin verilir.

EPOUTEN biti set edilmişse uçnokta OUT olarak kullanılabilir.

EPINEN biti set edilmişse uçnokta IN olarak kullanılabilir.

EPSTALL biti STALL kesmesi oluştuğunda(UIR.STALL) set olur.Hangi uçnokta STALL ile yanıt verdiyse bu bit sayesinde öğrenilir.Yazılımsal olarak temizlenmelidir.

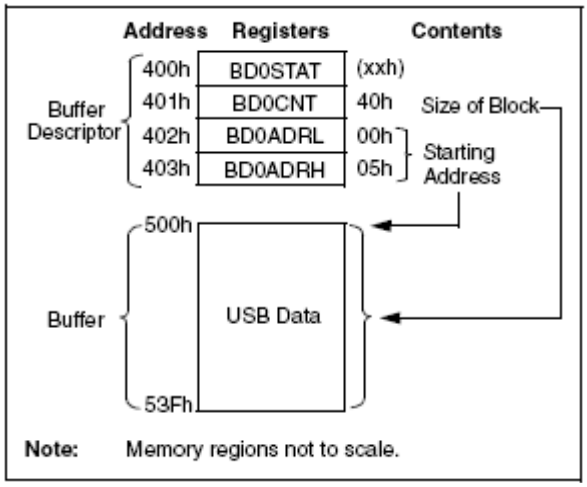
Bu kontrol register'larının yanında **UADDR** register'ı listeleme sırasında cihaz'a Set_Address isteği ile gönderilen adresi saklamak için kullanılır.**URFM** register'ı URFMH ve URFML yazmaçlarında 11 bitlik çerçeve numarasını saklar.

USB RAM Yerleşimi

PIC18F4550 2KB'lık RAM alanına sahiptir.RAM alanı toplam 15 banktan oluşur.Bank 4 ve 7 arası alan, SIE ve mikrokontrolör çekirdeği tarafından paylaşımlı olarak kullanılır.Bu alan 0x400 ve 0x7FF arası olmak üzere toplam 1Kb'dır.Bu büyüklükte bir alan USB'nin tüm işlemleri için yeterli büyüklüktedir.İzokron hariç diğer transferler full-speed modda maximum 64KB bloklar halinde bigi alıp gönderebilirler.İzokron transferlerde ise bu sınır 1024'dür.
0x400 ve 0x7FF arası USB Ram alanının, 0x400 ve 0x4FF arası olmak üzere toplam 256 byte'lık alanı Buffer Descriptor için kullanılır.Bu alan diğer işlemler içinde kullanılabilir.Şimdi Buffer Descriptor yapısını ve ne işe yaradığını inceleyelim.

Buffer Descriptor ve Buffer Descriptor Tablosu

Aşağıda Buffer Descriptor'ın PIC18F4550'nin datasheet'inden alınmış şekli verilmiştir.
Şekil-5) Tampon Künyesi



Yandaki şekilde görüldüğü gibi Buffer Descriptor yani tampon künyesi dört byte'lık bir alandan oluşmaktadır.
Tampon künyesi asıl kullanılacak olan bellek alanı hakkında SIE'ye bilgiler sağladığı gibi yonga kodu'na da bilgi sağlar.
BDnSTAT alanı ilerki bölümlerde detaylı bir şekilde incelenmiştir.
BDnCNT alanı gönderim işlemi yapılırken kaç byte bilgi gideceğini gösterirken, alım esnasında ise kaç byte bilgi okunduğunu gösterir.BDnADRL ve BDnADRH alanları ise bilgilerin tutulacağı tampon bölgesinin başlangıcına işaret etmektedir.
Buffer Descriptor alanı 16 adet uçnokta tarafından kullanılır. Yani her uçnokta'nın bir buffer descriptor'ı ve tamponu vardır.BDnCNT alanı uçnoktanın maximum paket büyüklüğüne göre yapılandırılır.Şekildeki örnekte BDnCNT alanına 0x40 yüklenerek uçnokta n tampon büyüklüğü, yani maximum paket büyüklüğü 64 olarak belirlenmiştir.

SIE veri alımı yaparken bu alandan daha büyük bir veri paketi alamaz.Çünkü uçnokta n maximum paket büyüklüğü 64 byte'dır.Bu işlem veri gönderimi içinde geçerlidir.İzokron transferlerde tek bir uçnokta kullanılmak şartı ile BDnCNT alanı 1024'e kadar çıkabilir.Böylece USB için ayrılan tüm bellek alanı bu uçnoktaya ayrılmış olur.
Fakat BDnCNT 1 byte olduğundan 1Kb alan tanımlaması için geriye kalan üst iki bit BDnSTAT alanının alt iki biti olan BC8 ve BC9 bitlerine yazılır.
Bu anlatılanları uçnoktaları, tanımlayıcıları ve PIC firmware'ini incelerken daha iyi anlayacaksınız.
BDnSTAT alanı CPU Mode ve SIE Mod olmak üzere iki şekilde kullanılır.Aşağıda BDnSTAT alanının her iki şekilde de açıklamaları bulunmaktadır.

BDnSTAT CPU Mod

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN(1)	DTS(2)	KEN	INCDIS	DTSEN	BSTALL	BC9	BC8
bit 7							bit 0

Şekil-6) Tampon Künyesi n Status Yazmacı (CPU Mod)

UOWN bit USB Buffer alanının CPU veya SIE tarafından kullanılmasını tayin etmek üzere basit bir semaphore mantığı ile çalışır.USB bellek bölgesi CPU çekirdeği ve SIE tarafından paylaşımlı olarak kullanıldığından bellek alanına aynı anda hem CPU hemde SIE erişemez.Bu bit'in 0 olması halinde Buffer sahibi (Buffer OwnerShip) CPU'ya aittir ve her türlü kontrol fonksiyonları kullanılabilir.Bu bit'in set olması halinde ise buffer alanı SIE tarafından kullanılır ve hiçbir şekilde CPU tarafından erişilemez.Böylece paylaşımlı bellek alanı güvenli bir şekilde kullanılır.UOWN biti set edilmiş, yani bellek SIE tarafından kullanılıyorken transfer işlemi tamamlandığında UOWN otomatik olarak temizlenir ve buffer sahipliği CPU'ya bırakılır.Tek istisna KEN bitinin set olmasında gerçekleşir.Bu durumda tampon şartsız olarak SIE tarafından kullanılır ve bırakılmaz.

DST biti veri senkronizasyonu için kullanılır.Bu bitin set olması Data1, 0 olması ise Data0 paketini gösterir. Aşağıdaki tabloda veri senkronizasyonu'nun nasıl sağlandığı gösterilmiştir.

OUT Packet from Host	BDnSTAT Settings		Device Response after Receiving Packet			
	DTSEN	DTS	Handshake	UOWN	TRNIF	BDnSTAT and USTAT Status
DATA0	1	0	ACK	0	1	Updated
DATA1	1	0	ACK	1	0	Not Updated
DATA0	1	1	ACK	0	1	Updated
DATA1	1	1	ACK	1	0	Not Updated
Either	0	x	ACK	0	1	Updated
Either, with error	x	x	NAK	1	0	Not Updated

Şekil-7) Data Senkronizasyonu

DTS değer değişim biti olarak kullanılır.DTS bitinin kullanılabilmesi için DTSEN bitinin set edilmiş olması gerekir.Değer değişim biti sayesinde veri kaybı olup olmadığı anlaşılır.IN ve OUT işlemleri için değer değişim bitleri PID(Paket ID) alanında taşınır.DATA0 0011, DATA1 ise 1011'dir.Yani tek değişen 3.bittir.
Değer değişim biti hem alıcı hemde verici tarafından takip edilir ve başlangıçta her iki taraftada DATA0 olarak kurulur.Veriyi alan alıcı taraf vericinin göndermiş olduğu değer değişim bitini kendisinin ki ile karşılaştırır.Eğer aynı ise kendi değer değişim bitinin durumunu değiştirir ve ACK yollar.ACK ile karşılaşan verici ise kendi değer değişim bitini de değiştirir.Böylece değer değişim bitleri her iki tarafda da aynı olmuş olur.Yani bir sonraki süreç DATA1 olacaktır.İzokron transferlerde değer değişim biti daima DATA0'dır.Çünkü haberleşme anında ACK gönderilmediği için verinin şartsız olarak alındığı kabul edilir.
Yukarıdaki şekilde DTS bitinin senkronlandığında ve yanlış paket alımı halinde UOWN ve TRNIF bitlerinin durumu verilmiştir.Yanlış veri alımı halinde veri senkronizasyonu gerçekleşmez.Bu durumda bellek bölgesi bırakılmaz ve SIE'de kalır.(Yeniden veri gelebilir diye UOWN set olarak kalır)Aynı zamanda ACK gönderilse bile bu TRNIF bitinin durumunu değiştirmez.Dolasıyla donanım kesmesi oluşmaz ve yonga kodunun verinin geldiğinden haberi olmaz.Alıcı kısım bir hata halinde ya da meşgul ise NAK yollar.Bu durumda UOWN ve TRNIF bitleri değişmeden kalır.
Alıcı taraf ACK yollar fakat verici bunu alamazsa veri, aynı değer değişim bilgisi ile tekrar gönderilir.Fakat bu kez değer değişim biti alıcı tarafından değiştirilmez.Değer değişim biti değişmediğinden, ve yeni gelen değer değişim biti eskisi ile aynı olduğundan her iki tarafta tekrar senkronlanır.

KEN biti bellek bölgesinin, veri alım işlemi bittikten sonra dahi SIE'de kalması için kullanılır.

INCDIS biti SIE'nin bellek bölgesine her yazımı sonrası bellek adresini otomatik olarak arttırması için kullanılır.Bu bit set edilirse adres otomatik olarak artar aksi halde artmaz.(Uçnokta SSP için)

DTSEN biti veri senkronizasyonunun olup olmayacağını belirler.Bu bit set edilirse veri senkronizasyonu var aksi halde yoktur.İzokron transferlerde bu bit temizlenebilir.

BSTALL biti genellikle kontrol transferlerde kullanılır ve desteklenmeyen istek, kontrol isteği başarısız veya uçnokta başarısız gibi işlemlerde uçnoktanın STALL ile yanıt verebileceğini belirler.
Bu şekilde kullanıma USB Spesifikasyon Protokol Stall adını vermiştir.Aynı zamanda uçnoktanın HALT özelliği bir Set_Feature isteği ile set edilmişse uçnokta veri alışverişi yapamaz duruma gelir ve her isteği STALL ile yanıtlamalıdır.Bu şekilde kullanılmasına Spesifikasyon Fonksiyonel Stall adını vermiştir.Yani BSTALL biti set edilirse istekler STALL ile karşılanır.

BC8-BC9 bitleri veri adedi BDnCNT alanını aşması halinde, üst bitlerin bu bitlere yazılması için kullanılır.Böylece BDnCNT maximum 1024 olabilir.

BDnSTAT SIE Mod

R/W-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	—	PID3	PID2	PID1	PID0	BC9	BC8
bit 7							bit 0

Şekil-8) Tampon Künyesi n Status Yazmacı (SIE Mod)

UOWN bit bu modda set olmalıdır.Bu bit hakkında bilgi CPU mod anlatılırken verilmiştir.

PID0-PID bitleri paket alımı sonrası Packet ID'leri gösterir.Bu bitler kontrol edilerek paket'in IN, OUT veya SETUP olduğu tespit edilebilir.

PING-PONG Buffer Özelliği

Bu özellik UCFG register'ının PPB0:PPB1 bitleri ile yapılandırılarak kullanılır.Bir uçnokta için ping-pong buffer tanımlanmışsa iki Buffer Descriptor girişi set edilir.Bunlar Even transfer ve Odd transfer'dir.Bu sayede CPU işlemleri için bir Buffer Descriptor verilecek, SIE işlemleri için ise diğer Buffer Descirptor verilecektir.
Bir uçnokta için tanımlanacak toplam dört adet Ping-Pong modu vardır;

- Ping-Pong desteklenmiyor
- Ping-Pong Buffering sadece Uçnokta 0 için destekleniyor
- Ping-Pong Buffering tüm uçnoktalar için destekleniyor
- Ping-Pong Buffering Uçnokta 0 hariç tüm uçnoktalar tarafından destekleniyor.

USB Modül, herbir uçnokta için Ping-Pong işaretçilerini izleme konumunda tutar.Bir işlemin tamamlanmasından sonra(UOWN biti SIE tarafından temizlenip, TRNIF biti set olduktan sonra) PPB işaretçisi toggle yaparak Odd BD'ye setlenir.Hemen ardından tamamlanan bir sonraki işlem'den sonra da tekrar toggle yaparak Even BD'ye döner.Ping-Pong Buffer işaretçi'nin Odd/Even BD durumu USTAT register'ının PPBI biti sayesinde takip edilir.Buffer Descriptor Table için tamponlama modları yerleşimi, PIC18F4550'nin 177.sayfasında verilmiştir.

USB Kesmeleri

USB Interrupt Status Register – UIR --

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
—	SOFIF	STALLIF	IDLEIF(1)	TRNIF(2)	ACTVIF(3)	UERRIF(4)	URSTIF
bit 7							bit 0

Şekil-9) USB Kesme Yazmacı

SOFIF biti SIE tarafından Start-Of-Frame jetonu alındığında set olur.Start-Of-Frame yani çerçeve başlangıcı jetonu ileriki bölümlerde incelenecektir.Bu bit yazılımsal olarak temizlenmelidir.

STALL biti SIE tarafından STALL gönderildiğinde set olur.Bu durumun gerçekleşmesi için ilgili uçnoktanın, tampon künyesi alanındaki BSTALL özelliğinin set olması gerekir.Bu bit yazılımsal olarak temizlenmelidir.

IDLEIF biti, cihaz bus'da 3ms boyunca faliyet görmediğinde set olur.Bu bit yazılımsal olarak temizlenmelidir.

TRNIF biti beklenen transfer tamamlandığında set olur.Yazılımsal olarak temizlenmelidir.

ACTVIF biti bus'da yeniden faliyet başladığında set olur.IDLEIF'in set olmasının ardından suspend duruma geçen bir cihaz bu bitin set olması durumunda suspend durumundan çıkar.

UERRIF biti UEIE bitlerinden biri daha önce kuruldu ise ve bir hata oluştu ise set olur.Yazılımsal olarak sadece temizlenebilir.

URSTIF biti bus'da reset durumu algılandığında set olur.Yazılımsal olarak temizlenmelidir.

UIE register'ı UIR register'ındaki kesmelerin oluşması için maskeleme bitlerini içerir.Bu yüzden ayrıca bit açıklamaları yapılmayacaktır.Ayrıntılı bilgi için datasheet'e bakın.
Aynı zamanda diğer hata kesmeleri gibi register'lar tasarım esnasında fazla kullanılmayacağından burada açıklanmamıştır.

USB Osilatör Konfigrasyonu Ayarları

PIC18F4550'nin USB işlemleri ve diğer işlemler için gelişmiş osilatör seçenekleri vardır.En göze çarpan özelliği ise full-speed USB transferini desteklediği için 48MHz saat freakansı ile çalışabilmesidir.
Aşağıda datasheet'den alınmış osilatör konfigrasyon tablosu yer almaktadır.

Input Oscillator Frequency	PLL Division (PLLDIV2:PLLDIV0)	Clock Mode (FOSC3:FOSC0)	MCU Clock Division (CPUDIV1:CPUDIV0)	Microcontroller Clock Frequency
48 MHz	N/A(1)	EC, ECIO	None (00)	48 MHz
			÷2 (01)	**24 MHz**
			÷3 (10)	16 MHz
			÷4 (11)	12 MHz
48 MHz	÷12 (111)	EC, ECIO	None (00)	48 MHz
			÷2 (01)	**24 MHz**
			÷3 (10)	16 MHz
			÷4 (11)	12 MHz
		ECPLL, ECPIO	÷2 (00)	48 MHz
			÷3 (01)	32 MHz
			÷4 (10)	**24 MHz**
			÷6 (11)	16 MHz
40 MHz	÷10 (110)	EC, ECIO	None (00)	40 MHz
			÷2 (01)	20 MHz
			÷3 (10)	13.33 MHz
			÷4 (11)	10 MHz
		ECPLL, ECPIO	÷2 (00)	48 MHz
			÷3 (01)	32 MHz
			÷4 (10)	**24 MHz**
			÷6 (11)	16 MHz
24 MHz	÷6 (101)	HS, EC, ECIO	None (00)	**24 MHz**
			÷2 (01)	12 MHz
			÷3 (10)	8 MHz
			÷4 (11)	6 MHz
		HSPLL, ECPLL, ECPIO	÷2 (00)	48 MHz
			÷3 (01)	32 MHz
			÷4 (10)	**24 MHz**
			÷6 (11)	16 MHz

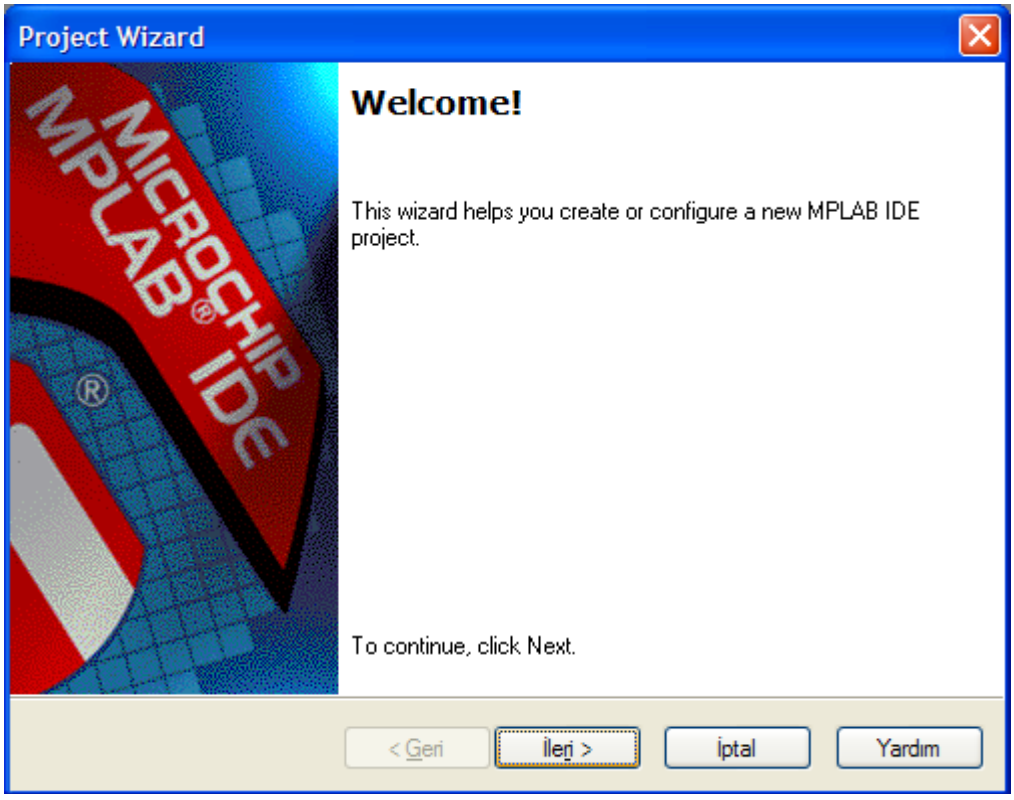
Input Oscillator Frequency	PLL Division (PLLDIV2:PLLDIV0)	Clock Mode (FOSC3:FOSC0)	MCU Clock Division (CPUDIV1:CPUDIV0)	Microcontroller Clock Frequency
20 MHz	+5 (100)	HS, EC, ECIO	None (00)	20 MHz
			+2 (01)	10 MHz
			+3 (10)	6.67 MHz
			+4 (11)	5 MHz
		HSPLL, ECPLL, ECPIO	+2 (00)	48 MHz
			+3 (01)	32 MHz
			+4 (10)	24 MHz
			+6 (11)	16 MHz
16 MHz	+4 (011)	HS, EC, ECIO	None (00)	16 MHz
			+2 (01)	8 MHz
			+3 (10)	5.33 MHz
			+4 (11)	4 MHz
		HSPLL, ECPLL, ECPIO	+2 (00)	48 MHz
			+3 (01)	32 MHz
			+4 (10)	24 MHz
			+6 (11)	16 MHz
12 MHz	+3 (010)	HS, EC, ECIO	None (00)	12 MHz
			+2 (01)	6 MHz
			+3 (10)	4 MHz
			+4 (11)	3 MHz
		HSPLL, ECPLL, ECPIO	+2 (00)	48 MHz
			+3 (01)	32 MHz
			+4 (10)	24 MHz
			+6 (11)	16 MHz
8 MHz	+2 (001)	HS, EC, ECIO	None (00)	8 MHz
			+2 (01)	4 MHz
			+3 (10)	2.67 MHz
			+4 (11)	2 MHz
		HSPLL, ECPLL, ECPIO	+2 (00)	48 MHz
			+3 (01)	32 MHz
			+4 (10)	24 MHz
			+6 (11)	16 MHz
4 MHz	+1 (000)	XT, HS, EC, ECIO	None (00)	4 MHz
			+2 (01)	2 MHz
			+3 (10)	1.33 MHz
			+4 (11)	1 MHz
		HSPLL, ECPLL, XTPLL, ECPIO	+2 (00)	48 MHz
			+3 (01)	32 MHz
			+4 (10)	24 MHz
			+6 (11)	16 MHz

Şekil-11) Osilatör Konfigrasyon Ayarları (Devam)

Biz uygulamalarımızı full-speed mod'da tasarlayacağımızdan 48MHz'e ihtiyacımız olacaktır.Bunu doğrudan OSC girişlerine uygulayabileceğimiz gibi 20MHz veya tabloda gösterilen diğer osilatör frekanslarını uygulayarak da kullanabiliriz.Fakat bunun için gerekli konfigrasyon ayarlarını bu duruma göre ayarlamalıyız.
PIC18F4550'nin dahili 96MHz PLL'i bulunmaktadır.Bu PLL içeride 2'ye bölünerek full-speed işlemler için (FSEN biti "1") 48MHz'lik saat kaynağı elde edilir.Fakat 96MHz'lik PLL giriş olarak daima 4MHz saat işaretine ihtiyaç duymaktadır.Bu durumda dışarıdan direkt 4MHz uygulayabileceğimiz gibi PLLDIV'de yapacağımız değişiklikle farklı osilatör kaynaklarınıda kullanabiliriz.
Diyelim ki full-speed modda USB tasarımı yapıyoruz ve 20MHz saat kaynağı kullanıyoruz.Bu durumda 96MHz'e 4MHz giriş elde etmek için PLL Prescaler'a "100" binary bilgisinin yüklenmesi gerekir.Bu durumda dışarıdan uygulanan saat sinyali 5'e bölünür ve 96MHz PLL için 4MHz'lik saat kaynağı sağlanmış olur.
Peki USB işlemleri için 48Mhz kullanılırken CPU frekansı kaç Mhz olacaktır? CPU saat kaynağı, 96MHz PLL çıkışından 48MHz'e dönüşmeden, yani ikiye bölünmeden alındığı için dahili CPU freakansı prescaler'ına 96MHz olarak gelecektir.Fakat bu freakans CPU'ya gitmeden CPUDIV PLL prescaler'a geldiği için CPUDIV "00" iken, bu saat işareti ikiye bölünerek CPU çekirdeğine gider.Yani USB 48Mhz kullanırken CPU saat kaynağıda 48Mhz olur.Fakat bu CPUDIV'e yüklenecek değerler ile değiştirilebilir.Örnek olarak 10 Mhz CPU frekansı kullanmak istiyorsak FOSC3:FOSC0 HS, CPUDIV1:CPUDIV0 "01" yani /2 olarak ayarlanmalıdır.Bu durumda dahili CPU saati 10Mhz olur.FOSC3:FOSC0 HS-PLL olarak ayarlanırsa, aynı ayarlar geçerli olmak üzere CPU frekansı 48Mhz olacaktır.Bu anlatılanlar yukarıdaki tablolarda gösterilmiştir.

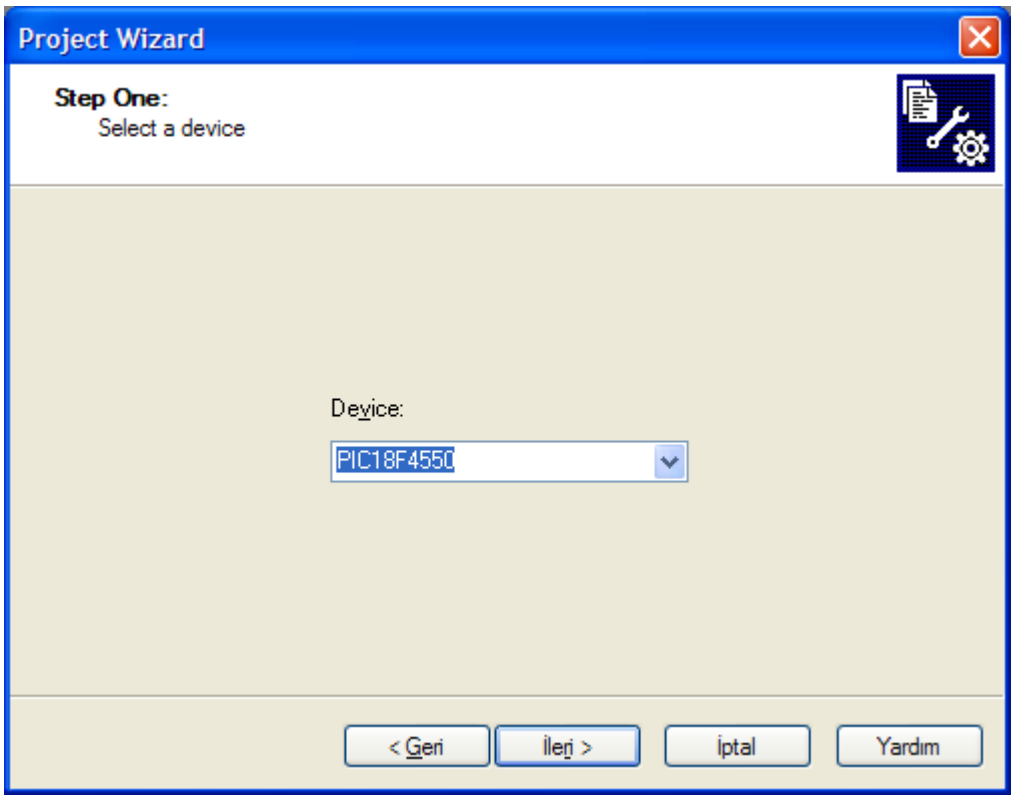
Derleyici Seçimi ve Programlama Dili

Bir mikroişlemci veya denetleyici ile çalışırken donanıma gerçekten hakim olmak için assembly dili kullanmak gerekir.Fakat assembly dili uzun ve karmaşık programlarda hem kod takibini zorlaştırır hemde bakımını ve geliştirilmesini engeller.Bu sebepten dolayı aynı assembly gibi donanıma hakim olabilecek, hemde program belleğinde az yer kaplayacak, hemde hemen hemen assembly kadar hızlı çalışacak bir dil seçmek en mantıklısı olacaktır.
Bu anlatılanlar doğrultusunda seçilebilecek tek dil C dilidir.Bizde projelerimizi geliştirirken C dilini kullanacağız. Microchip firması'nın 18xxx serisi denetleyicileri için geliştirdiği komut seti C için optimize edilmiştir ve sadece 18xxx serisi işlemciler ile kullanılacak C18 derleyicisi bulunmaktadır.
C18 derleyicisini kurmak için *StepByStepUSBTools\PIC Araçlar\C18* dizini altından C18Compiler üzerine çift tıklayın ve kurulumu başlatın.Bu C18'in 1.1 sürümüdür.Kurulum işlemi tamamlandıktan sonra aynı dizin içindeki Upgrade'i çift tıklayın ve kurulumu tamamlayın.Bu sayede C18 derleyici sürümü son versiyona yükselmiş olacaktır.Eğer bilgisayarınızda MPLAB kurulu değil ise *StepByStepUSBTools\PIC Araçlar\MPLAB* dizini altından son sürümü kurabilirsiniz.
Şimdi yeni bir proje oluşturmak için basit olarak hangi adımlar izlenmeli bunlara bakalım.Bu makalenin amacı MPLAB ve C18 yazılımını incelemek olmadığından, bu konuya taraflı bir şekilde değinmek yerine basit olarak bir proje nasıl oluşturulur ve derlenir bunları inceleyelim.
İlk olarak MPLAB programını açın.*Project* menüsünden *Projet Wizard* 'ı başlatın.Karşınıza aşağıdaki gibi bir pencere çıkacaktır.



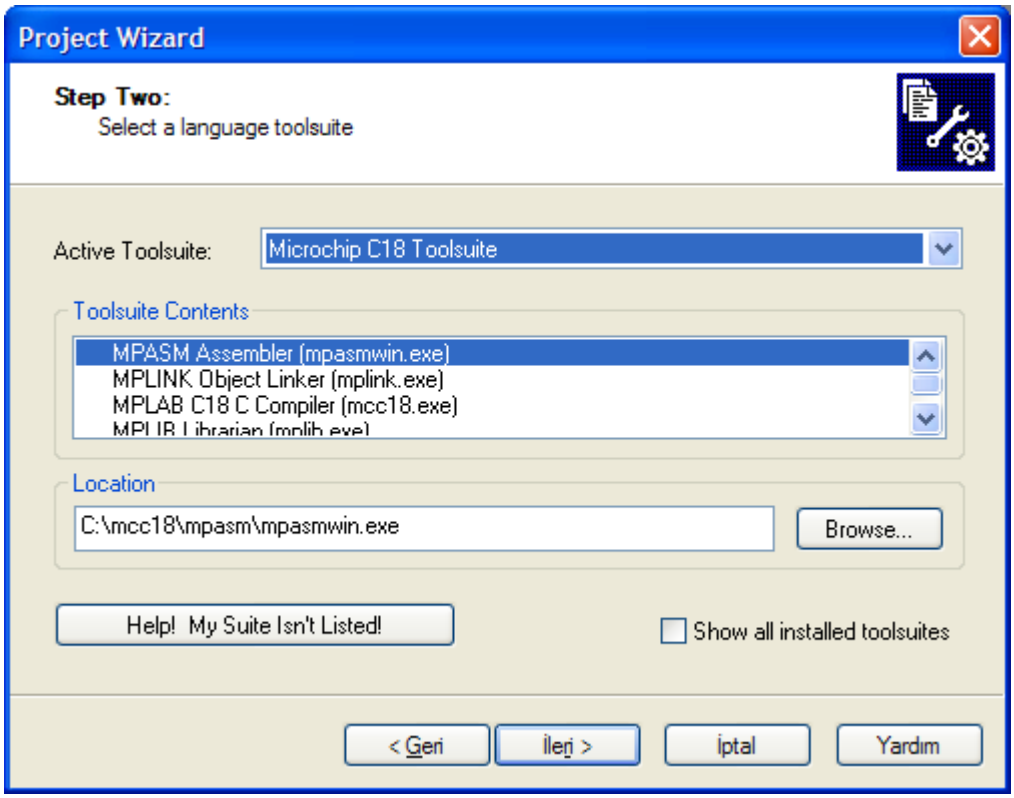
Şekil-12) Project Wizard Karşılama Ekranı

Buradan İleri butonuna tıklayın ve bir sonraki aşamaya geçin.Karşınıza aşağıdaki gibi bir pencere çıkacaktır.Bu pencereden hangi PIC'le çalışmak istiyorsak bu PIC'i listeden seçmeliyiz.Biz uygulamalarımızda PIC18F4550'yi kullanacağımız için bu denetleyiciyi seçin.



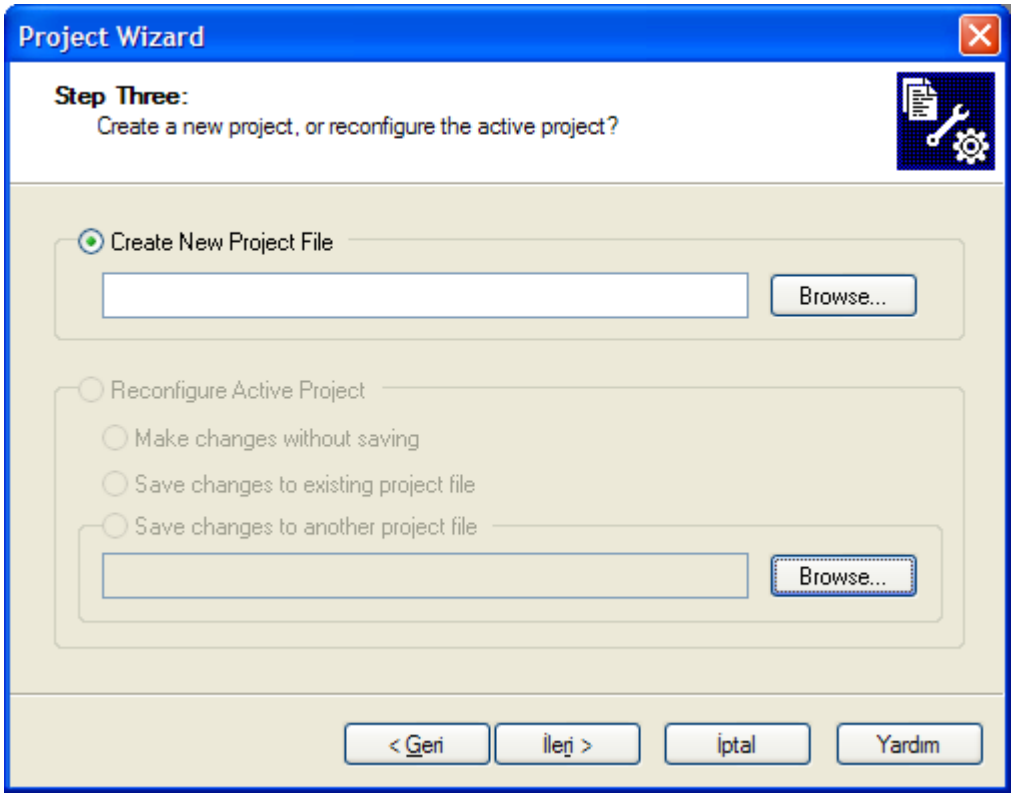
Şekil-13) Denetleyici seçme penceresi

Denetleyici seçimini yaptıktan sonra ileri butonuna tıklayın.Karşınıza hangi derleyici ile çalışmak istediğinizi seçebileceğiniz bir pencere çıkacaktır.Bu pencerede MPLAB ile kullanılabilecek derleyiciler yer almaktadır.MPASM'yi seçerek assembly ile derleme yapabilirsiniz.Fakat biz C18 ile derleme yapacağımızdan, sizin aşağıdaki pencerede görülen ayarları yapmanız gerekmektedir.



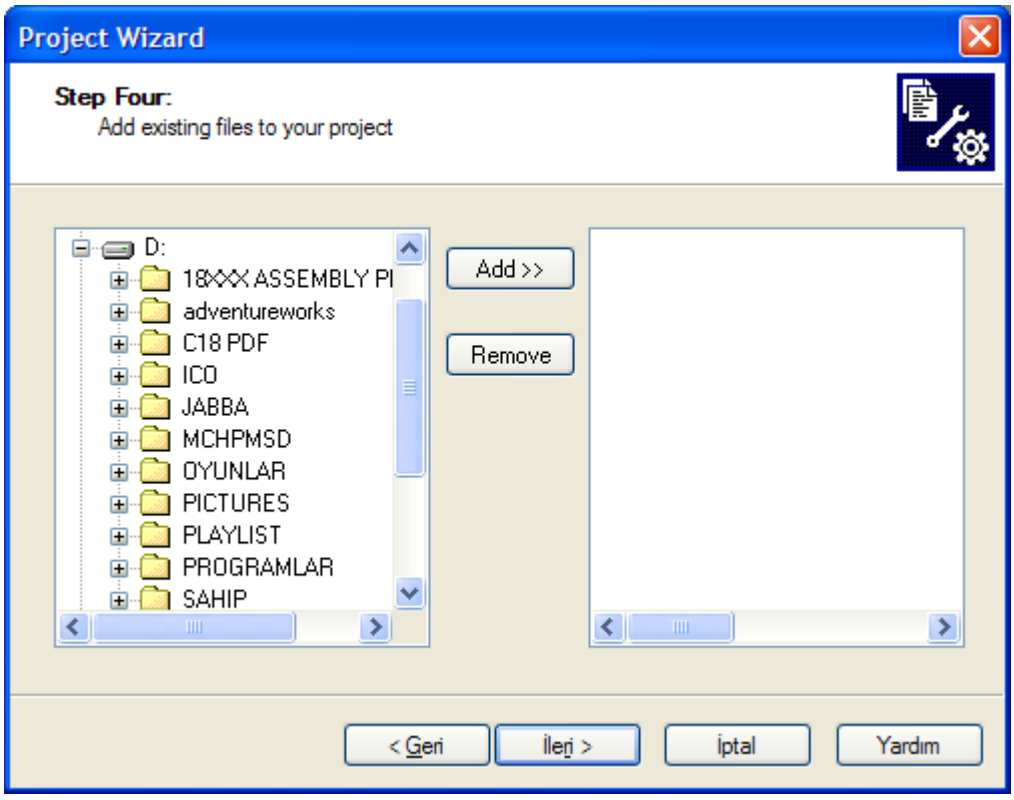
Şekil-14) Derleyici seçme penceresi

Bu adımı da tamamladıktan sonra, projenizin oluşturulacağı dizini belirlemek için aşağıdaki gibi bir pencere ile karşılaşacaksınız.

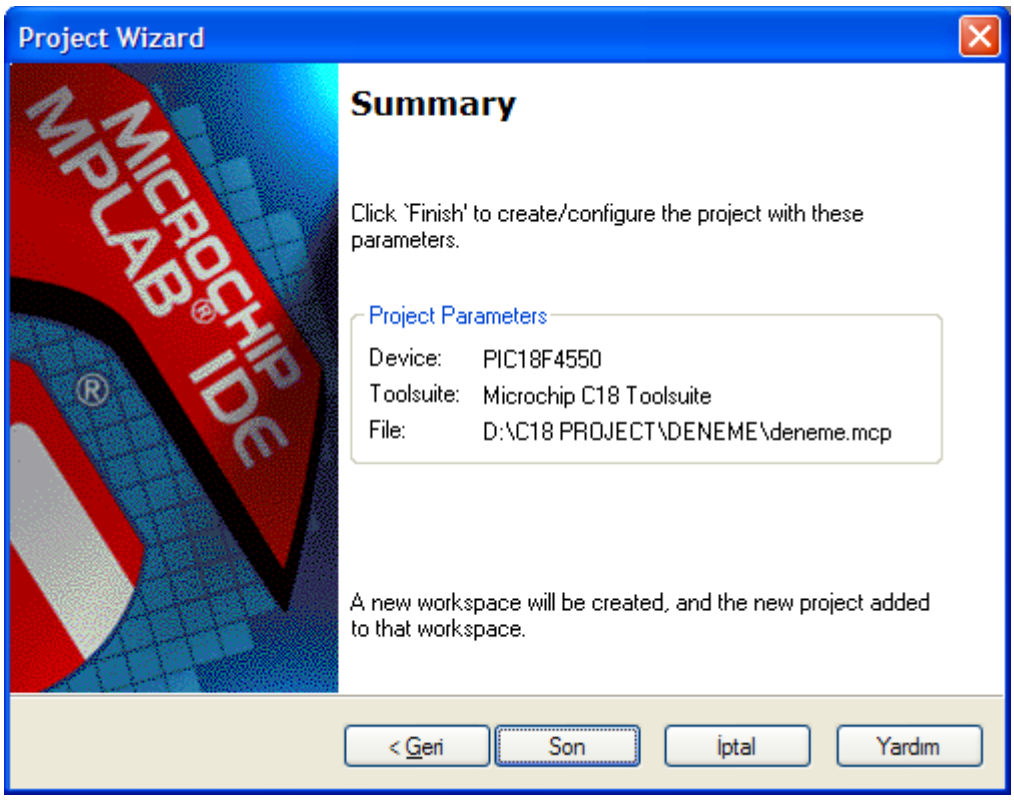


Şekil-15) Proje dizini seçme penceresi

Bu pencerede, projenin oluşturulacağı dizini elle girebileceğiniz gibi *Browse* butonuna tıklayarak açılan pencereden de seçebilirsiniz.Proje dizini belirlenirken, projenize bir isim vermeniz istenecektir.Gerekli gördüğünüz ismi verdikten sonra *İleri* butonuna tıklayarak bir sonraki aşamaya geçin.Karşınıza projelerinize herhangi bir dosya ekleyip-kaldırabileceğiniz aşağıdaki gibi bir pencere gelecektir.

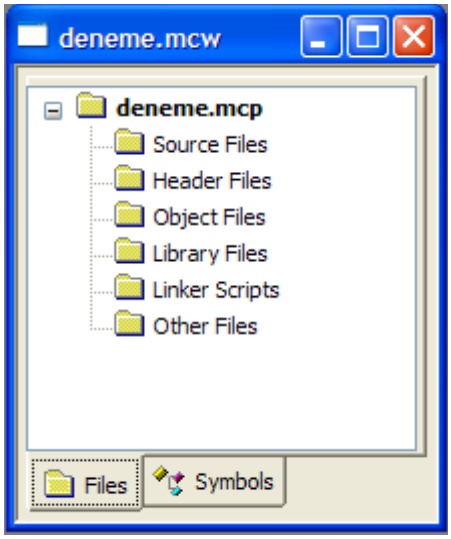


Bu pencereden örneğin bir PDF dosyasını projenize ekleyebilirsiniz.Son olarak tüm bu işlemler tamamlandıktan sonra proje tasarımı bitecek ve aşağıdaki gibi pencere gösterilecektir.



Şekil-16) Project Wizard'ın tamamlandığını gösteren pencere

Proje tasarımı bittikten sonra karşımızda aşağıdaki gibi bir pencere olmalı.Bu pencereden yeni kaynak dosyaları eklenebilir veya kaldırılabilir.



Şekil-17) Proje Alanı

Örneğin *File* menüsünden *New* seçeneğini seçin.Karşınıza yeni bir kod sayfası çıkacaktır.Bu sayfaya aşağıdaki gibi basit bir kod yazın.

```
#include <p18f4550.h>

void main( void )
 {
      PORTCbits.RC0 = 1;
      for(;;);
 }
```

Şimdi bu kodları *File/Save* seçeneklerini seçerek projenizi oluşturuduğunuz dizine kaydedin.Ardından yandaki resimde görülen **Source Files** simgesine sağ tıklayın ve *Add Files'a* tıklayın.Karşınıza çıkacak arama penceresinden dosyasınızı bulun ve ekleyin.Şimdi yapılacak tek şey derleme işlemi için gerekli Linker dosyasını eklemek.

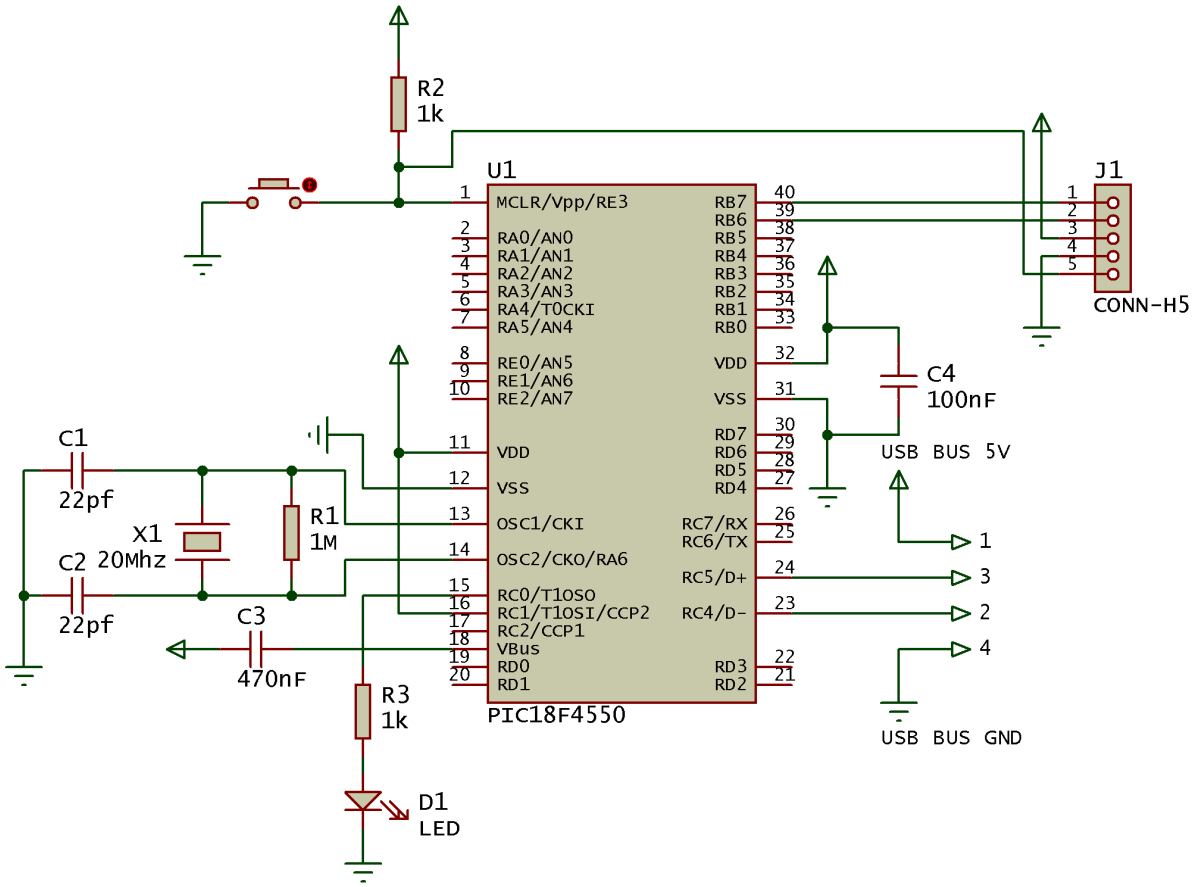
Linker dosyası parçalara ayrılmış dosyalardaki fonksiyonların adreslerinin ve değişken adreslerinin bilinmesi ve diğer dosyalar tarafından kullanılabilmesi için gereklidir.Bu dosyayı eklemek için **Linker Scripts** simgesine sağ tıklayın ve çıkan pencereden *C:\mcc18\lkr* dizinine gidin.Buradaki linker dosylarından 18f4550_i.lkr adlı dosyayı bulun ve ekleyin.Artık proje derlenmeye hazır durumda.Şimdi *Project* menüsünden Build All seçeneğini seçin.Proje, eğer eksik birşey yapılmadıysa hatasız derlenecektir.

Programlayıcılar

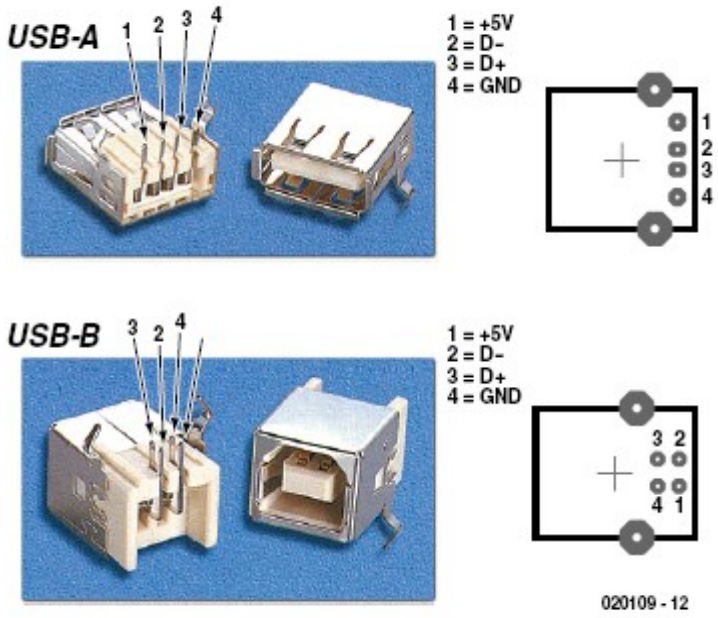
PIC ile tasarlanacak uygulama için yazılmış gömülü kodun yüklenebilmesi için bir programlayıcıya ihtiyaç vardır.Bunun için internetten birçok devre şeması bulunabilir.Fakat eğer PIC programlama ile uğraşıyorsanız sıfırdan bir programlayıcı hazırlamak vakit kaybından başka bir şey değildir.Bu iş için piyasada hazır olarak satılan birçok programlayıcı kart bulunmaktadır.Üstelik bunların fiyatı alacağınız PIC denetleyici ile hemen hemen aynı fiyattır.
Kendiniz yapmaktansa hazır bir programlayıcı satın almanız, hem sizi meydana gelecek hatalarla uğraşmaktan kurtaracak hemde buna ayırdığınız zamanı kod yazmak için kullanabileceksiniz.Tabiki bu bahsedilenler orta seviyeli PIC'ler için geçerli.16FXXX serisi denetleyiciler için programlayıcılar düşük fiyatlarda olabiliyorken 18FXXX serisi gibi denetleyiciler için hazırlanmış programlayıcı kartlar daha pahalı olabiliyor.
Biz uygulamalarımız boyunca Microchip firmasının bir ürünü olan ICD2 programlayıcısını kullanacağız.Bu programlayıcının en önemli özelliği, MPLAB programının desteği ile devre üzeri hata ayıklama imkanı olmasıdır.(In-Circuit Debuging)Bu programlayıcı sayesinde yazdığınız kodları adım adım çalıştırabilir, PIC içindeki register'ların gerçek değerlerini eş zamanlı olarak takip edebilirsiniz.
ICD2'nin bir diğer avantajı ise geniş bir aygıt listesine sahip olmasıdır.Yani bu programlayıcı ile Microchip firmasının neredeyse tüm PIC'lerini programlayabilirsiniz.Bunun en büyük getirisi ise herhangi bir PIC ile çalışırken başka model bir PIC ile çalışmanız gerektiğinde programlıyıcı devresini değiştirmenize gerek kalmamasıdır.ICD2'yi satın almak isteyenlerin yaklaşık olarak 250-300$ + KDV gibi bir fiyatı gözden çıkarmaları gerekiyor.Yinede bu programlayıcıyı satın alacak güce sahip olmayanlar için *PIC Araçlar/Programlayıcılar* dizini altında bazı devre şemaları verilmiştir.Bu devre şemaları denenmedikleri için çalışıp çalışmadıkları test edilmelidir.

Test Devresi Ve Özellikleri

Aşağıda bu makale boyunca uygulamalarımızı geliştireceğimiz temel devre'nin şeması verilmiştir.Bu devreyi delikli plakete kurabilirsiniz.Diğer devreleri yaparken gerekli bağlantıları bu devre üzerinden alacağımızı düşünürsek sağlam olmak şartı ile ya bread board'a veya delikli plakete kurmak daha mantıklı olacaktır.



Bu devrede CONN-5 isimli header ICD2'nin bağlantı kablosunun takıldığı yerdir.ICD2 haricinde bir programlayıcı ile çalışanların, bu kısmı kurmaları gerekmez.
USB porta bağlanması gereken uçlar harici olarak gösterilmiştir.Sırası ile gösterilmiş 1,2,3,4 numaralı pinler USB konnnektörünün hangi bacaklarına bağlanması gerektiğini gösterir.Aşağıda A tipi ve B-tipi mini USB konnektörler için pin numaraları gösterilmiştir.



Şekil-17) Pin tanımlamaları

Uygulama devresi besleme gerilimini USB portundan almaktadır.Fakat Motor Kontrol ve diğer benzeri uygulamaların beslemelerini bu devreden ayrı bir besleme kaynağından almak gerekmektedir.

Windows Ugulaması İçin Programlama Dili

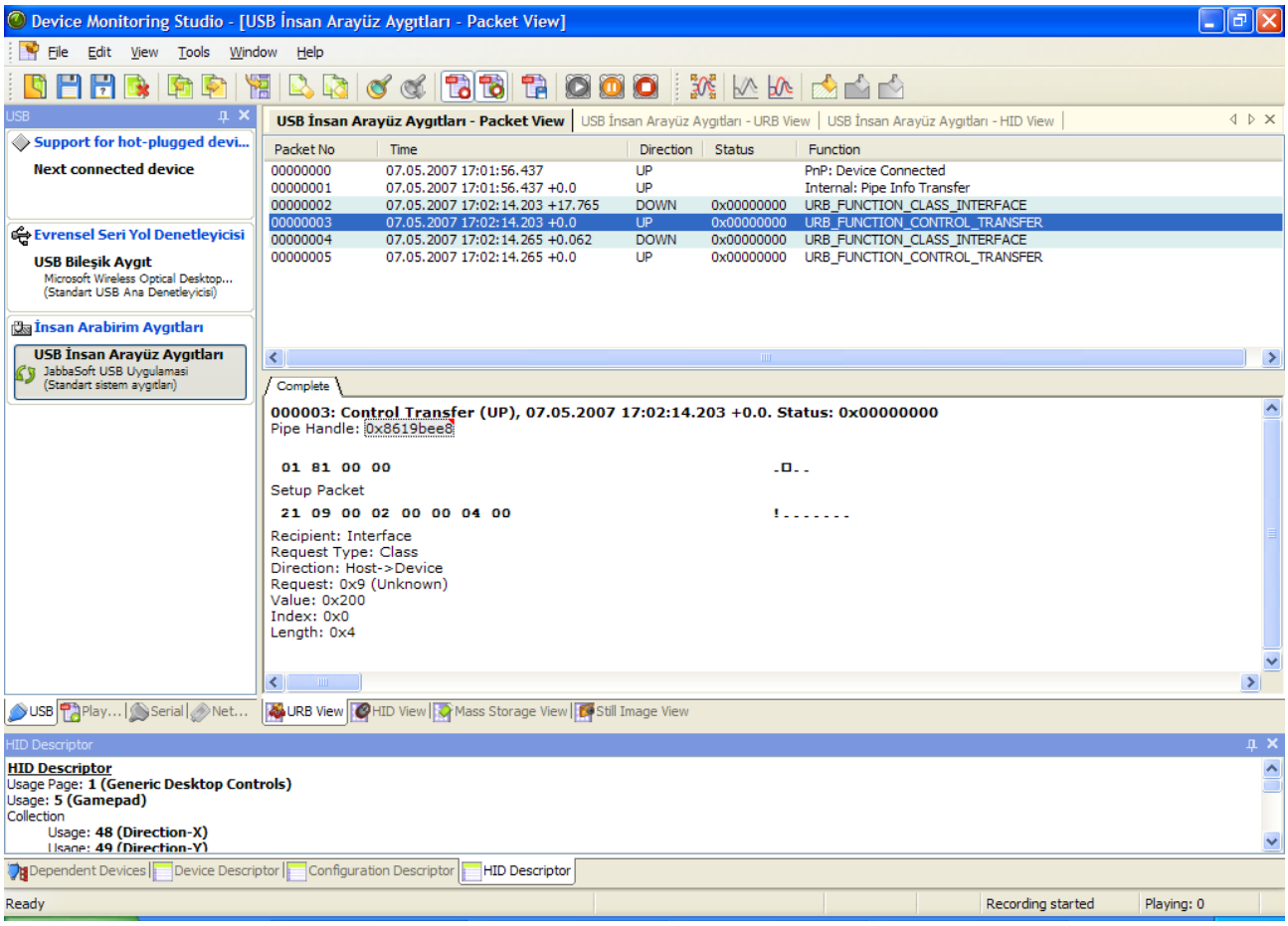
Bu makale boyunca tüm uygulamalarda DOTNET dillerinden birisi olan C# kullanılmıştır.Bu sebepten dolayı ek içerisinde verilmiş uygulamaları çalıştırabilmek ve kod dosyalarını açıp inceleyebilmek için bazı işlemler yapmamız gerekmektedir.İlk olarak C# bir DOTNET dili olduğundan bilgisayarınıza .NET Framework 2.0 veya daha yüksek bir sürümünü kurmanız gerekmektedir.NET Framework 2.0 ve 3.0 sürümlerini *StepByStepUSB-Tools/DOTNET* dizini altında bulabilirsiniz.Bu paketi yükledikten sonra artık uygulama programları çalıştırılabilir.Fakat uygulama programlarının kaynak kodlarını da incelemek için bilgisayarınıza Visual Studio 2005 IDE'sini kurmanız gerekmektedir.Eğer bilgisayarınızda bu IDE kurulu değil ise Microsoft'un resmi sitesinden Visual C# 2005 Express Edition'ı indirip kurabilirsiniz.

USB Projesi Geliştirirken Kullanılacak Test Programları

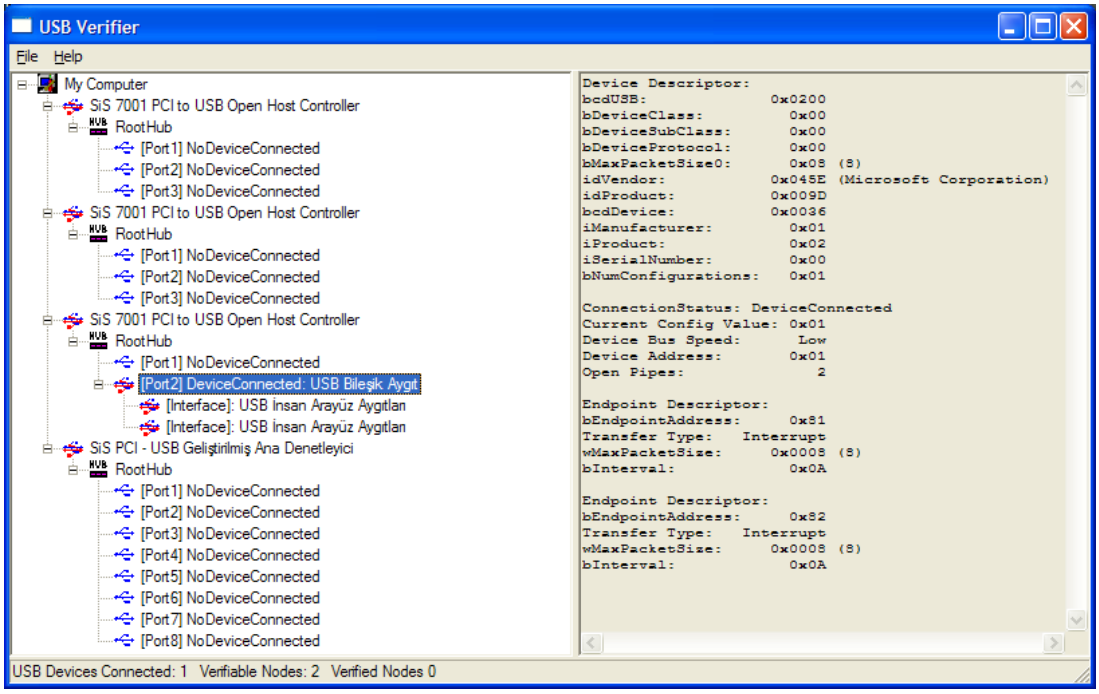
Device Monitoring Studio

USB protokolü diğer protokollere göre çok daha zordur.Sistem doğru tasarlandıysa kusursuz çalışacağı kesindir, fakat eğer biryerde ufak bir hata yapılmışsa işler son derece karmaşık bir hal alır.Bu yüzden hem hataların giderilmesinde, gelen-giden paketlerin, elsıkışmaların izlenmesinde tam donanımlı bir test programının olması işleri daha da kolaylaştıracaktır.Bu sayede hem haberleşme izlenerek USB protokolünün detayları daha rahat anlaşılabilecek, hemde olası hatalar daha kolay tespit edilebilecektir.
Biz uygulamarımız boyunca iki test aracı kullanacağız.Birincisi Device Monitoring Stdio, ikincisi ise USB Verify aracıdır. Device Monitoring Studio programının 14 günlük trial versiyonunu *StepByStepUSBTools/USB Test*

Araçlar dizininden bulup kurabilirsiniz.Bu programın full versiyonu ücretli olduğundan detaylar için programın üreticisine ait siteyi ziyaret etmelisiniz.USB Verify programı ise ücretsizdir.Bu programıda aynı dizinde bulabilir ve kurmadan direkt çalıştırabilirsiniz.Aşağıda bu programların arayüzlerinin resimleri verilmiştir.Program hakkındaki detaylar ise ilerleyen bölümlerde bitirdiğimiz tasarımı test ederken verilecektir.



Şekil-17) Device Monitoring Studio programı



Şekil-17) USB Verify Programı

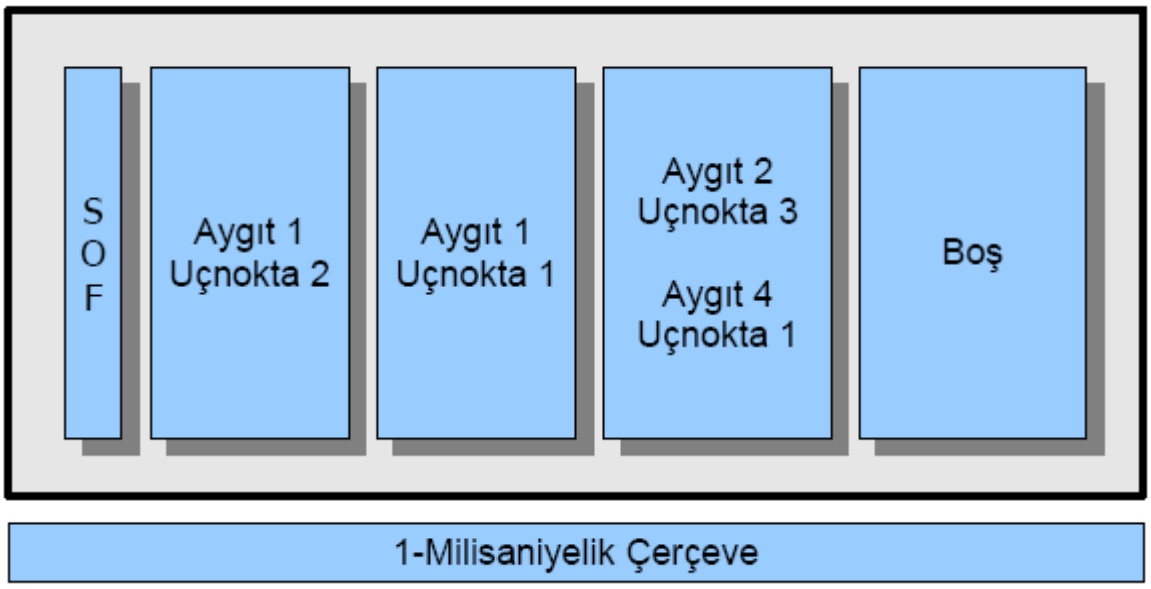
Bölüm 3

USB Transferi'nin Detayları

USB haberleşmesi daha öncede söylendiği gibi diğer protokollere göre daha karmaşık bir yapıya sahiptir.Seri veri iletişimi gibi bir haberleşme türünde bir start biti, ardından bir byte'lık veri ve stop biti gönderilir.Bu kolay ve basit bit haberleşmedir.Oysa ki USB haberleşmesinde işlemler paketler halinde gerçekleşir.Bir işlemde bir veya birden fazla veri paketi, hata bitleri ve el sıkışma bitleri gibi değerler gönderilir ve alınır.
USB donanımı ise yine diğer protokollerden farklıdır.USB'nin tek bir veri yolu bulunur ve tüm cihazlar bu veri yolunu kullanırlar.Oysa ki seri portta her portun veri yolu birbirinden ayrıdır.USB portları tek bir veri yolu üzerinden windows ile haberleşirler.Bus tüm cihazlar'a Windows tarafından eşit zaman aralıklarınca paylaştırılır.Şimdi bu veri trafiğini daha detaylı bir şekilde inceleyelim.

Bus'taki Verinin Yönetimi

USB'nin iki adet sinyal hattı bulunur.Bunlar D+ ve D- pinleridir.Buradaki + ve – sembolleri bir hattan high diğer hattan low taşınıyor şeklinde algılanmamalıdır.Bu ifadelerin anlamları 10.Bölümde **Sinyaller ve Şifreleme** başlığı altında incelenmiştir.Bu iki sinyal hattı tüm cihazlar tarafından paylaşılır ve veri zıt yönlerde taşınır.BUS, tüm cihazlar tarafından eşit zaman aralıklarında paylaşılabilmesi için çerçeve adı verilen parçalara bölünür.Bu çerçeve düşük ve tam hızlı cihazlarda 1ms'dir.Her işlemin bir veya birkaç paketten oluştuğunu belirtmiştik.Eğer bir işlem bir çerçeve süresinde bitmeyecek kadar büyük ise bu işlem birkaç çerçeveye bölünür.Yüksek hızlı cihazlarda çerçeve sekiz parçaya bölünür ve mikroçerçeve olarak adlandırılır.Her çerçeve SOF denilen(Start-Of-Frame) bir zamanlama referansı ile başlar.Aşağıda 1ms'lik çerçeve şekli görülmektedir.



Şekil-18) 1ms'lik Çerçeve

Her çerçeve SOF paketi ile başladıktan sonra bunu takiben cihaz ve uçnokta adresleri ile birlikte veri işlemleri ile devam eder.Çerçeve içindeki işlemler PC tarafından istediği şekilde düzenlenebilir.Tüm cihazlar BUS'ı ortak olarak kullandıklarından her işlemde verinin varış adresi ve diğer bilgiler yer almak zorundadır.Cihaz kendi adresini içermeyen işlem gördüğünde göz ardı edecek, adres'i belli cihaz bu işlemden payını alacaktır.Her cihazın listeleme sırasında PC tarafından atanmış bir adresi bulunur.Bu yüzden her işlemde yukarıdaki şekilde görüldüğü gibi cihaz ve uçnokta adresleri bulunmak zorundadır.
Bilgi transferini her zaman PC başlatır.Bir cihaz herhangi bir anda herhangi bir veriyi istediği gibi PC'ye gönderemez.Her işlem PC tarafından belirlenen isteklerden oluşur ve cihaz sadece bu isteklere yanıt verir.

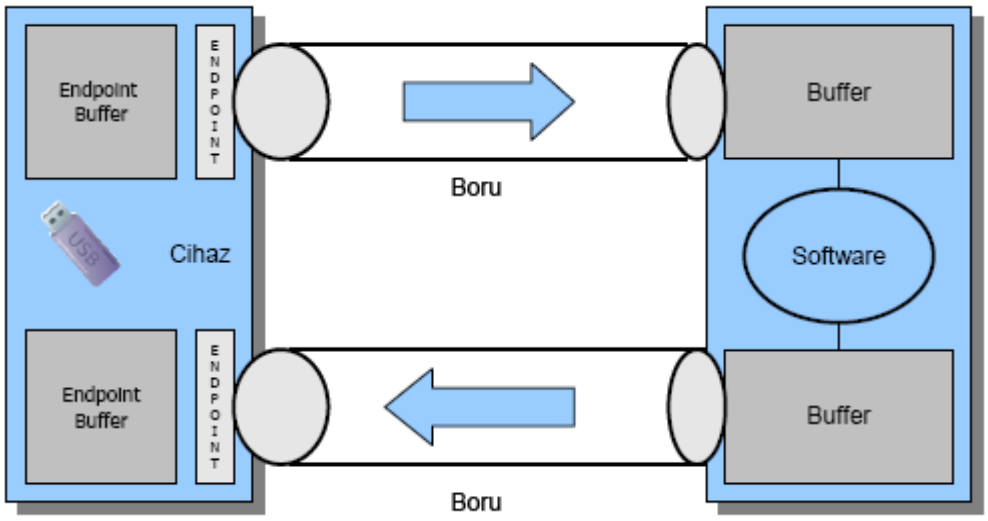
Transfer Elemanları

Uçnokta Nedir?

Uçnokta, denetleyici içindeki bir yazmaç olabileceği gibi birçok byte'ı barındıran bir tampon da olabilir.PC cihaz ile haberleşmede uçnoktaları kullanılır.Uçnoktalarda ya PC'den gelmiş veriler ya da gönderilmeyi bekleyen veriler bulunur.PC uçnoktalar ile haberleşmede başlangıç noktasıdır fakat PC'nin uçnoktaları yoktur.Bunun yerine veri gönderim ve alım işlemlerinde tampon kullanılır.Her uçnoktası veriyi tek bir yönde taşır.Yani bir uçnokta ya veri alır (OUT) ya da veri gönderir.(IN)
Bir uçnoktanın çift yönlü olması özel bir durumdur.Örneğin kontrol transferlerinde kullanılan bir uçnokta hem IN hemde OUT olabilir.Bir uçnokta adresi o uçnoktanın numarasıdır ve bu değer 0x00 ile 0x0F arası olabilir.(IN ve OUT olmak üzere toplam 16)IN ve OUT olarak çift yönlü yapılandırılmış bir uçnokta her yönde tek bir adresi kullanır.
Her cihaz Uçnokta0'ı desteklemek zorundadır.İlave uçnoktalar tercihe bağlıdır.Uçnokta0 default olarak kontrol transferleri için gereklidir.IN ve OUT olduğundan çift yönlü iletişim yapar.Sadece kontrol uçnoktası bulunduran bir cihaz kontrol işlemleri sonrası(listeleme) veri alışverişini daha sonra gösterilecek yöntemlerle, yine kontrol transferlerine başvurarak yapar.Haberleşme sırasındaki her işlem veri akışının yönünü gösteren bir kod içerir.Bu kodlar IN, OUT ve SETUP'dır.Bunlar ileriki bölümlerde detaylı bir şekilde anlatılacaktır.IN ile cihaz PC'ye veri gönderir, OUT ile veri alır.Bu ilgili uçnokta sayesinde gerçekleşir.IN uçnoktası cihaz'a bilgi göndermede OUT uçnoktası ise bilgi almada kullanılacaktır.PIC18F4550'de her uçnokta bir yazmaca karşılık gelir ve kendisine ait bir tamponu vardır.Gelen veriler bu tampona yazılır.Giden veriler de aynı şekilde ilgili uçnoktanın ilgili tamponuna yazılır.Daha öncede belirtildiği gibi bu tampon BDnADRL, BDnADRH yazmaçlarının belirtmiş olduğu adres bölgesinden başlar ve BDnCNT kadardır.

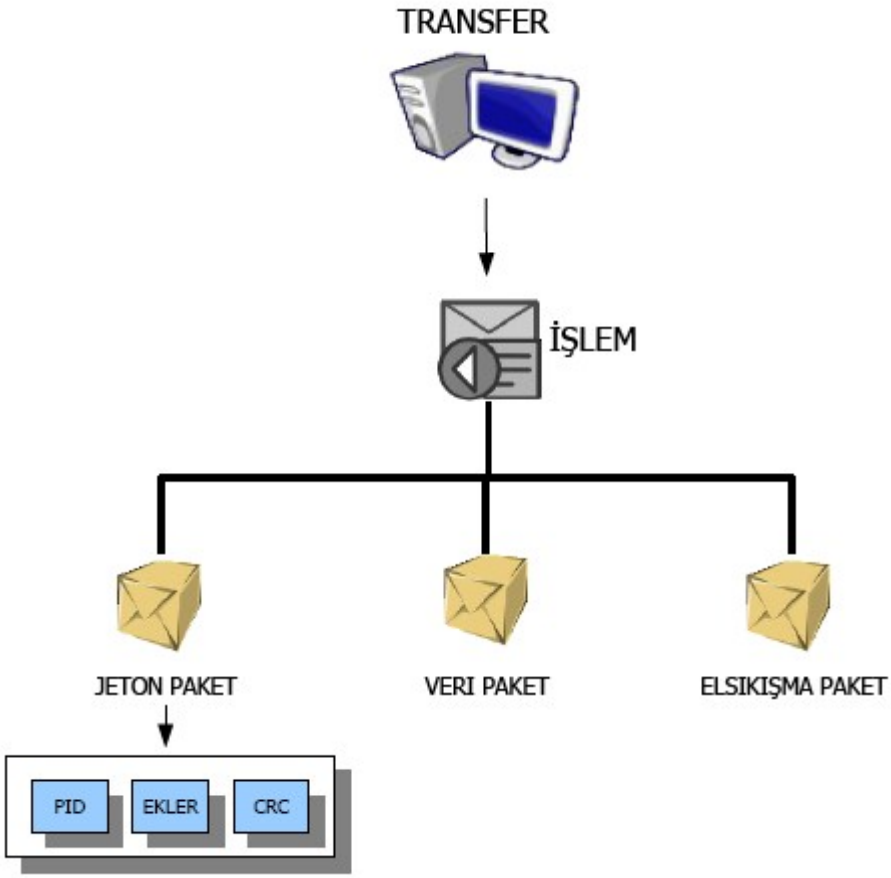
USB Borusu Nedir?

Cihaz ile PC arasındaki haberleşme, PC'nin listeleme sonrası uygun bulduğu cihazlar için döşediği "Borular" sayesinde gerçekleşir.Bu borular cihaz uçnoktası ile PC'deki denetleyici yazılımı arasında bir köprü oluşturur.
Bir cihaz listelendikten ve gerekli boruları döşendikten sonra, sistemden kaldırıldığında bu borular yine PC'deki denetleyici tarafından otomatik olarak kaldırılır.Bunun yanısıra herhangi bir anda PC'deki denetleyici yeni borular döşeme yoluna gidebilir veya mevcut boruları kaldırabilir.
Yapılandırma esnasında(listeleme) PC'ye her uçnokta için bazı tanımlayıcılar gönderilir.Tanımlayıcılar ilerki bölümlerde açıklanacaktır.Bu tanımlayıcılar uçnokta adresi, uçnokta yönü, desteklediği transfer tipi, maximum paket boyutu gibi bilgileri içerir.İşte bu bilgiler doğrultusunda PC tarafından gerekli borular uçnokta ile haberleşmek üzere döşenir.Borular fiziki nesneler değil yazılımsal nesnelerdir.
Bazı durumlarda boruların döşenmesi bantgenişliği etkeni yüzünden engellenebilir.Yeterli bantgenişliğini temin edemeyen PC boru döşemez ve bağlantı isteğini reddeder.Bu durumda cihaz gerekli bant genişliği sağlanana kadar bekleyebilir veya daha düşük bir bant genişliği konfigrasyonu ile yeniden bağlantı isteğinde bulunabilir.
Aşağıdaki resimde uçnokta ve borular arasındaki ilişki gösterilmektedir.



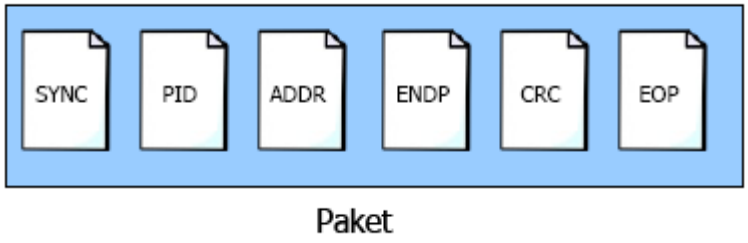
Borular Akış ve Mesaj olmak üzere ikiye ayrılır.Bilgi akışının çift yönlü olması halinde Mesaj boruları kullanılır.Örnek vermek gerekirse Kontrol transferler mesaj borularını kullanırlar.Diğer bütün transferler akış borularını kullanırlar.

Paket Tipleri ve İçerikleri



Şekil-20) İşlem Evreleri ve Paketler

Her transfer, transferin türüne göre birkaç basamaktan(işlem)oluşur.Bunlar SETUP, VERI ve STATUS basamaklarıdır.Yukarıdaki resimde bu basamaklardan sadece biri gösterilmiştir.Görüldüğü gibi her basamakta en az üç paket gönderilir.Bu paketler JETON, VERI ve ELSIKIŞMA paketleridir.Yine şekilde görüldüğü gibi her paket bir Paket tanımlayıcısı(Packet ID), ek veriler ve hata kontrol bitleri içerir.Aslında her paketin genel bir yapısı vardır ve PID alanındaki değerlere göre tanımlanırlar.Aşağıdaki resimde genel bir paketin içeriği gösterilmektedir.



Şekil-21) Bir paket'in yapısı

Şimdi yukarıdaki resimde gösterilen, paket içerisindeki alanları tek tek inceleyelim.

SYNC

Tüm paketler Sync ile başlar.Sync alıcı ve verici senkronizasyonunu sağlamak için gönderilir.Düşük ve tam hızlı cihazlarda 8 bit, yüksek hızlı cihazlarda ise 32 bit uzuluğundadır.Sync bitiminin hemen ardından PID başlar. Tam hızlı cihazlarda örnek verilirse 8 bitlik SYNC, KJKJKJKK şeklindedir.(Bu sinyaller daha sonra incelenecektir.)Idle Bus'dan ilk K'ya geçiş start biti gibi algılanır ve yeni paket geldiğini gösterir.

PID

Packet ID,gönderilen paket'in tipini tanımlamada kullanılan 8 bitlik bir değerdir.Alt 4 bit paket tipini gösterirken üst 4 bit ise alt bitlerin tümleyenidir ve hata kontrolünde kullanılır.JETON, VERI, ELSIKIŞMA ve özel paketler için 16 adet PID tanımlıdır.Aşağıdaki tabloda bu değerler verilmiştir.

GRUP	PID Değeri	Açıklama
JETON	0x01	OUT Jeton
	0x09	IN Jeton
	0x05	SOF Jeton
	0x0D	SETUP Jeton
DATA	0x03	DATA0
	0x0B	DATA1
	0x07	DATA2
	0xFF	MDATA
ELSIKIŞMA	0x02	ACK
	0x0A	NAK
	0x0E	STALL
	0x06	NYET
SPECIAL	0x0C	PREAMBLE
	0x0C	ERR
	0x08	SPLIT
	0x04	PING

Tablo-1) Packet ID Değerleri

Şekil-20'deki resimde görüldüğü gibi her işlem basamağında en az üç paket gönderiliyordu.Bu paketler şekil-21'deki yapıdadır.Bu paketlerin JETON, VERI ve ELSIKIŞMA paketleri olduğunu PID alanı belirler.

ADDR

Bu alan cihaz adres'ini içerir.Cihaz adresleri 0-127 arasında olabilir.

ENDP

Bu alan 4 bit uzunluğundadır ve uçnokta numarasını gösterir.4 bit uzunlukla maximum 16 uçnokta seçilebilir(Her biri çift yönlü olmak üzere 32)Düşük hız cihazlar yalnızca 2 uçnokta destekler.

CRC
Bu alan çevrimsel artıklık kontrolü (CRC) bitlerini içerir.Tüm jeton paketlerinde 5 bit, veri paketlerinde ise 16 bittir.

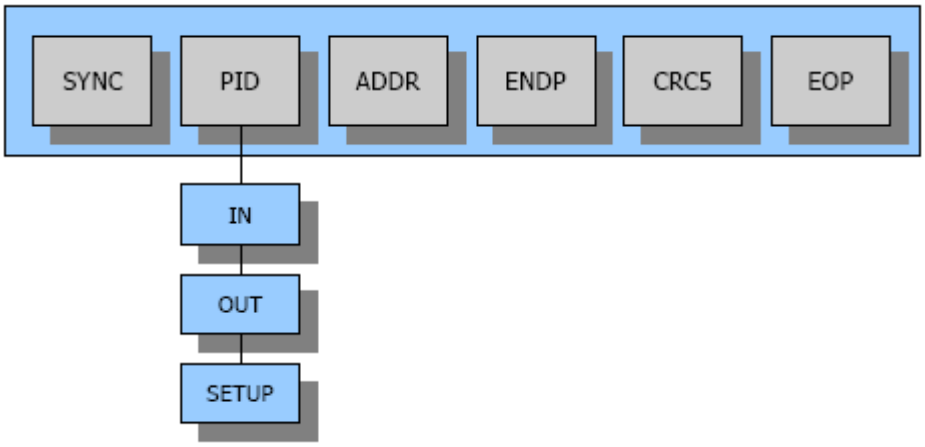
EOP
Bu sinyal bus'ı Idle mod'a sokar.(Aylak)Böylece bus sonraki SYNC için hazır olmuş olur.Düşük ve tam hız cihazlar için iki bit genişlikte olabilir ve kendinden referanslıdır.

Şimdi genel bir paket'in içeriğini öğrendiğinize göre paket tiplerini inceleyelim.SETUP, VERI ve STATUS işlem basamaklarının her birinde en az üç paket gönderilir.Bunlar JETON, VERI ve ELSIKIŞMA paketleridir.Bu paketlerin yapısı Şekil-21'deki gibi olmakla birlikte PID alanı, paketin tipini(JETON, VERI, ELSIKIŞMA) belirler ve veri paketinde isteğin içerdiği bilgiler gönderilir.Bu anlatılanlar kontrol transferleri incelenirken daha iyi anlaşılacaktır.Şimdi bu üç paketin yapısını inceleyelim.

JETON PAKET

Jeton paketi üç tipte olabilir.Bu PID alanındaki değerle belirlenir;

- **IN** – PC'ye VERI basamağında bilgi gönderileceğini ifade eder.
- **OUT –** VERI basamağında PC'den veri alınacağını ifade eder.
- **SETUP** – Kontrol transferlerde kullanılır.

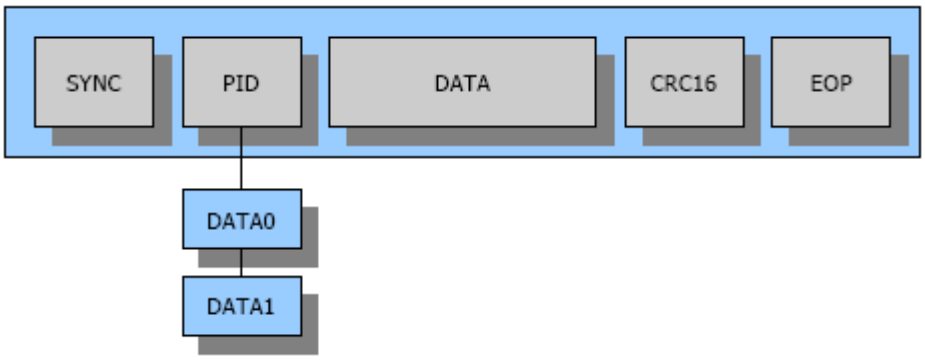


Şekil-22) JETON Paketi

DATA PAKET

Veri paketi iki tip olabilir.Bu PID alanındaki değerle belirlenir.Bu paket ile düşük hızlı cihazlar 8 byte, tam hızlı cihazlar 1023 byte ve yüksek hızlı cihazlar ise 1024 byte veri gönderebilir.

- **DATA0**
- **DATA1**

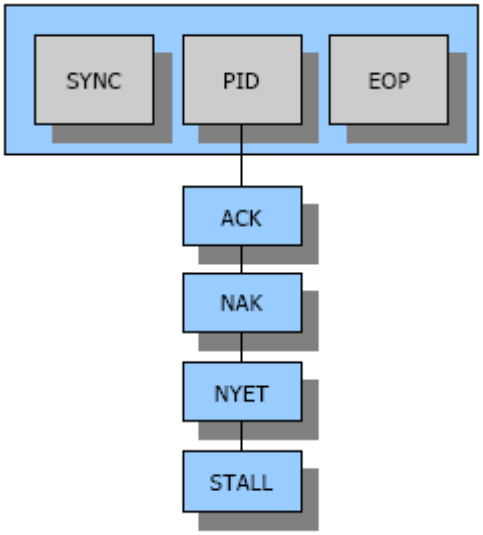


Şekil-23) Veri Paketi

ELSIKIŞMA PAKET

Elsıkışma paketi, alıcı tarafından gönderilir ve işlemin başarısını gösterir.Elsıkışma paketi dört tipte olabilir.

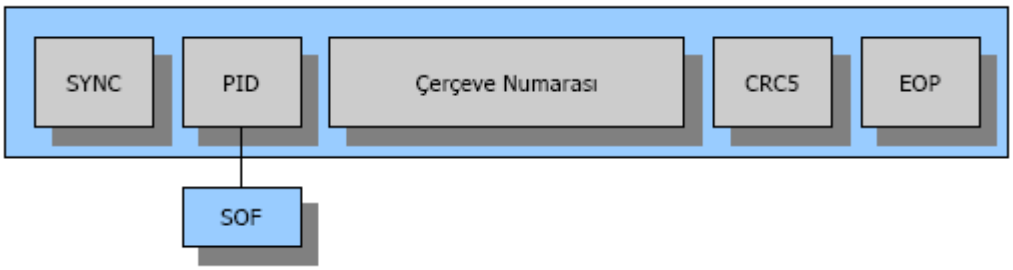
- **ACK** – Verilerin başarılı bir şekilde alındığını gösterir.
- **NAK** – Gönderecek veri yok demektir.PC bu durumda yeniden istekte bulunur.
- **STALL** – Desteklenmeyen istek, uçnokta çakılması gibi durumları gösterir.
- **NYET** – Veri alınabileceğini fakat uçnoktanın hazır olmadığını gösterir.



Şekil-24) Elsıkışma Paketi

START OF FRAME PAKET

Bu paket 11 bitlik bir çerçeve numarası içerir.Tam hız cihazlarda 1ms'de bir, yüksek hız cihazlarda ise 125us'de bir gönderilir.



Şekil-25) Start Of Frame Paketi

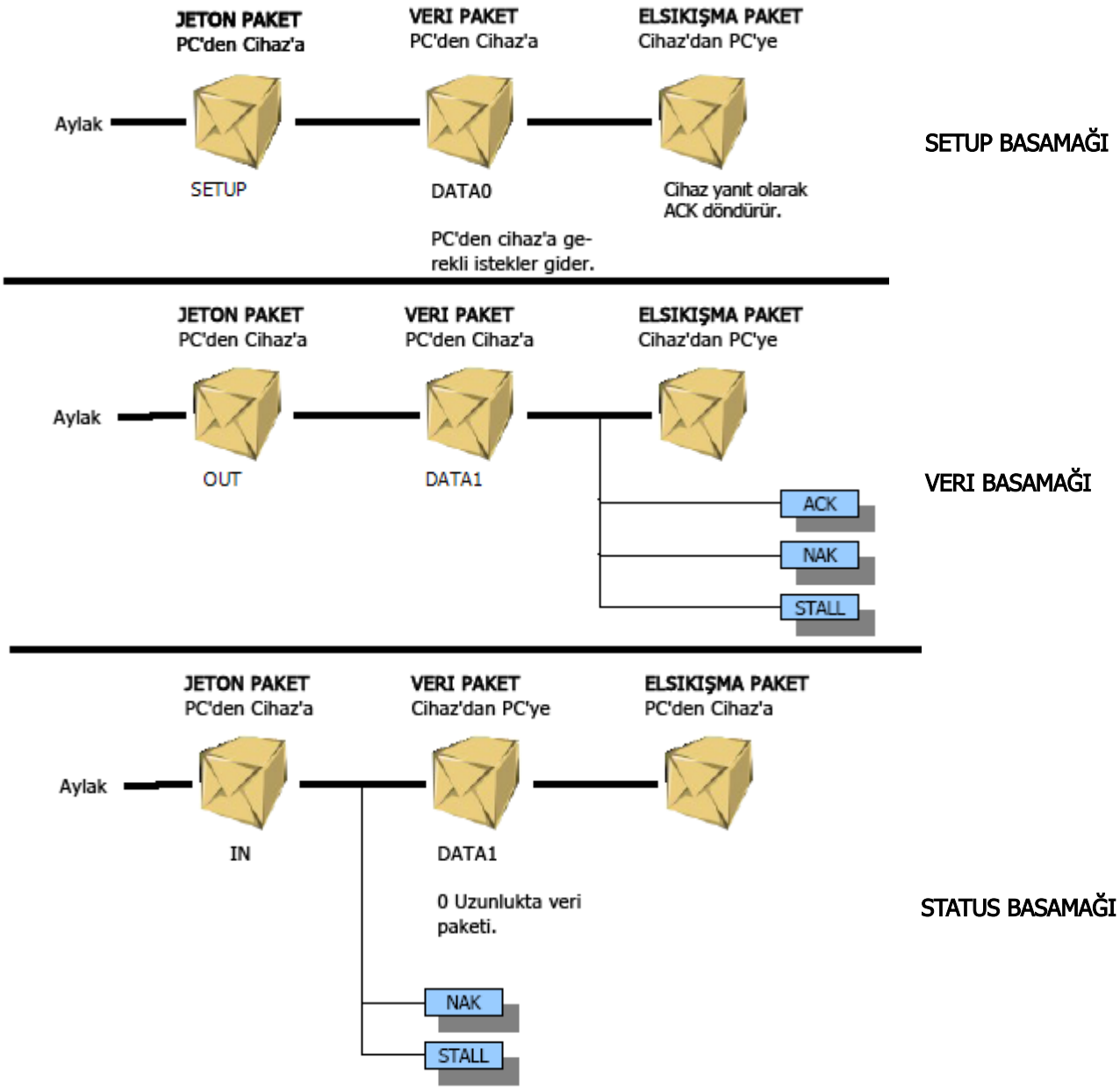
Transfer Türleri ve Özellikleri

Kontrol Transfer

Kontrol transferleri cihazların yapılandırılması, veri isteklerinin yürütülmesi için kullanılan, aynı zamanda blok veri iletiminide destekleyen bir transfer türüdür.Her USB cihazının varsayılan uçnoktası, yani uçnokta 0 üzerinden kontrol transferini desteklemesi şarttır.Çünkü PC'ye yeni bir cihaz takıldığında PC bu cihazın özelliklerini öğrenmek, gerekli bilgileri almak ve cihazı yapılandırmak için kontrol transferlerine baş vurur.Bu süreç listeleme sürecidir ve Aygıt Yöneticisi tarafından yürütülür.Kendini sorunsuz bir şekilde ifade etmiş bir cihaz Aygıt Yöneticisi tarafından listeye eklenir.Bir cihazın kontrol transferleri için yapılandırılmış ek uçnoktaları olabilsede çoğu zaman buna ihtiyaç yoktur.

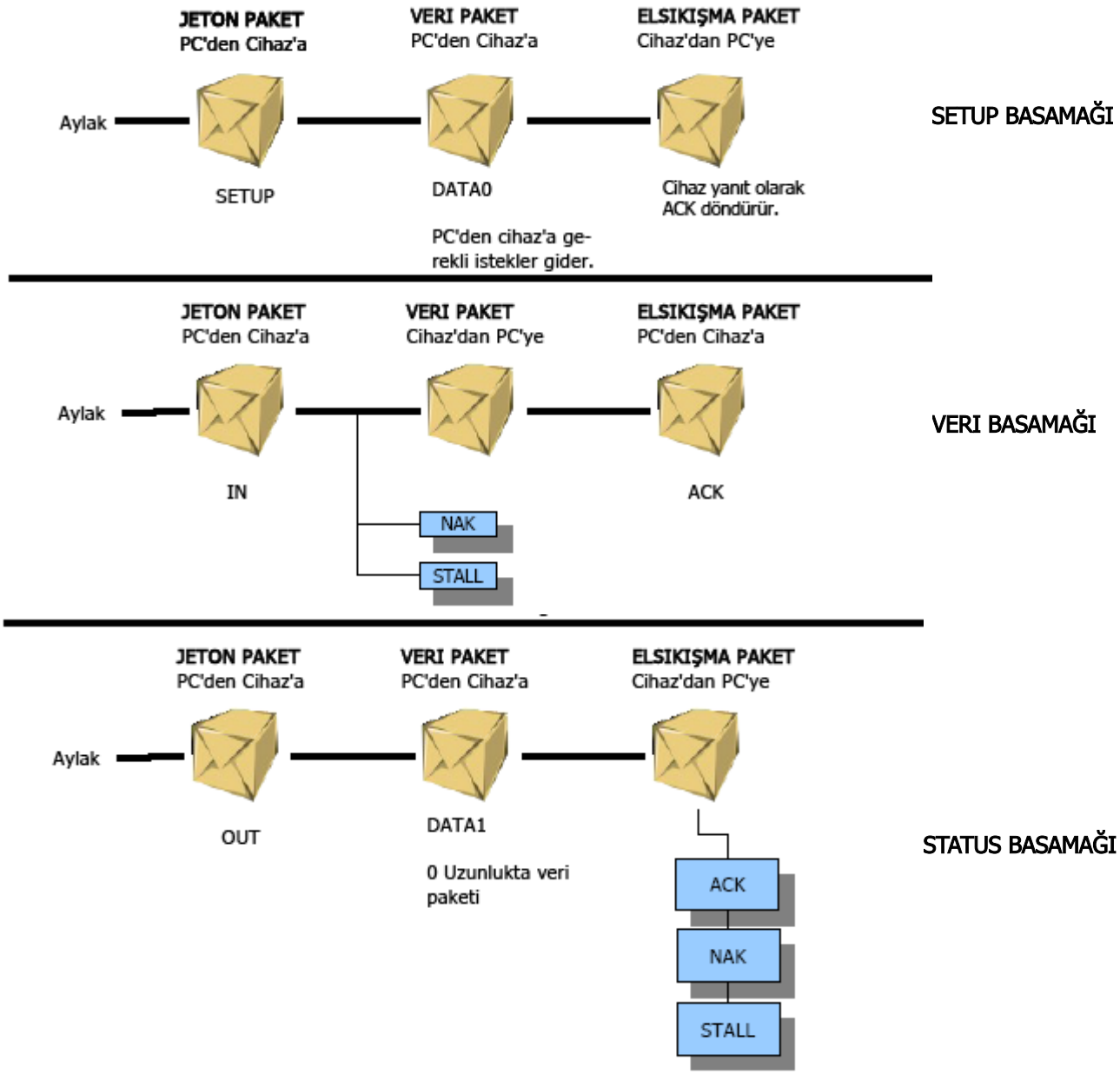
Kontrol transferlerinin üç basamaklı yapısı SETUP, VERI ve STATUS basamaklarından oluşur.Her basamakta en az üç paket iletilir.Bunlar JETON, VERI ve ELSIKIŞMA paketleridir.Bu paketlerin yapısı daha önce incelenmişti. Her kontrol transferinde SETUP ve STATUS basamaklarının olması zorunluluğu vardır.VERI basamağı ise her durumda gerekmediği gibi seçimliktir.Çünkü bazı kontrol transferlerinde isteğin gerektirdiği veri SETUP basamağının VERI paketinde gönderilebilmektedir.
Kontrol transferlerinde, bu iş için yapılandırılacak uçnoktası hem IN hemde OUT transferleri desteklemelidir.Çünkü kontrol transferinin mesaj borusu, bu uçnoktanın IN ve OUT işlemleri için aynı adresini kullanır.
Kontrol trasferlerinde cihazın bilgi alması Control Write, bilgi göndermesi ise Control Read işlemleri ile gerçekleşir.Şekil-26) da bu işlemler ayrıntılı olarak gösterilmiştir.
Ayrıca bir önceki paket tipleri ve içerikleri Kontrol Transfer'ler baz alınarak açıklanmıştır.

Control Write



Şekil-26) Control Write Transfer

Control Read



Şekil-27) Control Read Transfer

Control Write transferde cihaz PC'den bilgi almaktadır.PC ilk olarak SETUP basamağını başlatır.Burada JETON paketi gönderilir ve bu paketin PID alanı SETUP'dır.Böylece cihaz SETUP basamağının başladığını algılar.Bu işlemin hemen ardından VERI paketi gönderilir.Bu paket'in PID alanı DATA0'dır ve isteğin gerektirdiği bilgileri içerir.Cihaz tüm bu paketleri sorunsuz şekilde aldıysa ELSIKIŞMA paketi gönderir.Bu paketin PID alanı ACK'dir ve PC verilerin sorunsuz alındığını algılar.SETUP veya VERI basamaklarında bir hata tespit eden cihaz ELSIKIŞMA göndermez.
Son ACK gönderiminden sonra SETUP basamağı tamamlanmıştır.Böylece PC veri basamağını başlatır.Burada yine JETON paketi gönderilir.Eğer SETUP basamağında verilerin yönünün PC'den cihaz'a doğru olacağı belirtilmişse PID alanı OUT aksi halde IN olacaktır.PID alanı IN olduğu taktirde, JETON paketinin hemen ardından cihaz VERI paketi gönderir.Bu paketin PID alanı DATA1 olup isteğin içerdiği bilgilerin bir kısmını veya tamamını taşır.VERI paketinde gönderilen bilgileri sorunsuz alan PC bir ELSIKIŞMA paketi gönderir.Bu paketin PID alanı'da ACK'dir.Böylece VERI basamağı da tamamlanmış olur.PID alanı OUT olduğu taktirde PC cihaz'a veri paketini gönderir ve veriyi alan cihaz ise ELSIKIŞMA paketinde ACK ile yanıt verir.VERI basamağının tamamlanmasının ardından PC STATUS basamağını başlatır.PC bunun için bir JETON paketi gönderir.

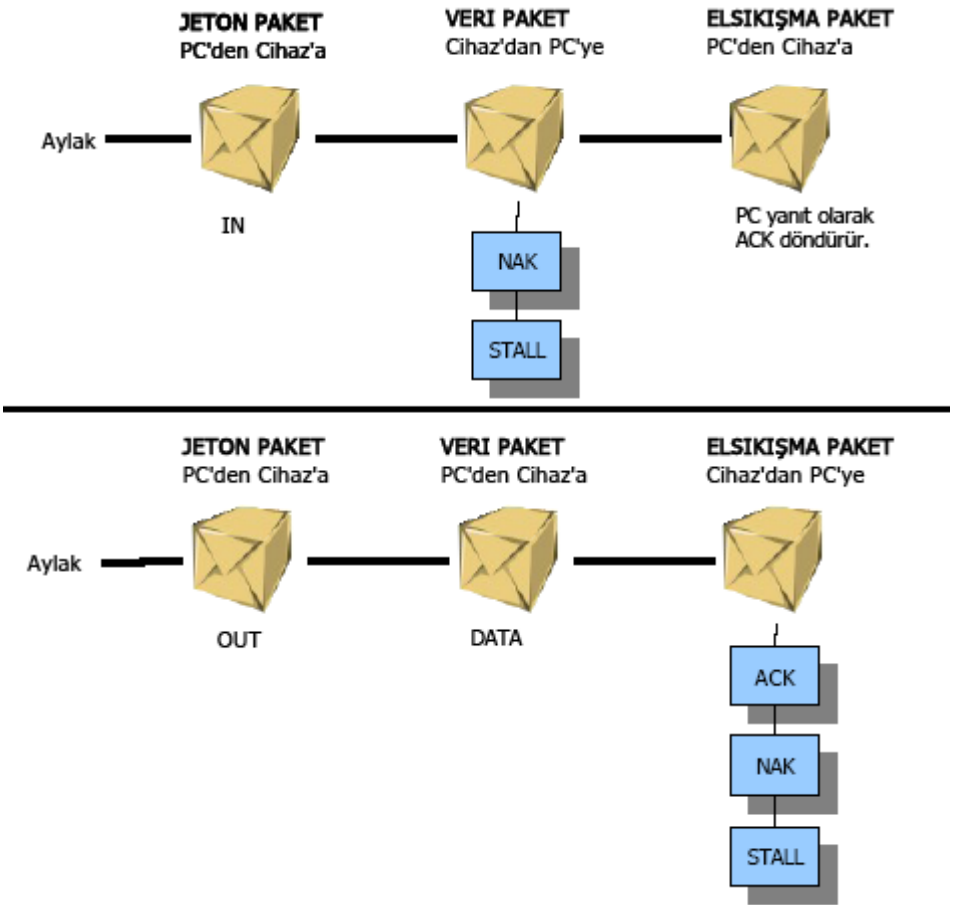
Bu paketin PID alanı bir önceki basamağın VERI paketinin alıcısının ters yönündedir.Yani VERI paketini cihaz almışsa(OUT) bu alanın PID alanı IN, eğer PC almışsa OUT olacaktır.
PID alanı IN ise cihaz, PC'ye sadece PID alanı ve hata kontrol bitleri olan sıfır uzunlukta bir VERI paketi gönderir.Bunun yanında ACK, NAK ya da STALL da gönderilebilir.PC ise ELSIKIŞMA paketi göndererek VERI paketini aldığına dair onay verir.PID alanı OUT ise PC cihaz'a sadece PID alanı ve hata kontrol bitleri olan sıfır uzunlukta VERI paketi yollar.Paketi alan cihaz ELSIKIŞMA paketi ile yanıt verir.STATUS basamağının tamamlanmasının ardından transfer tamamlanmış olur.

NOT: STATUS basamağı tüm transferin başarısını gösterir.STATUS bilgisi bu basamağın VERI paketinde taşınır ve sıfır uzunlukta bir paket veya ACK,NAK,STALL gibi değerler içerir.Sıfır uzunlukta bir paket işlemin sorunsuz tamamlandığını gösterir.

Control Read transferlerdeki mantık ise Control Write transferlerden farklı değildir.Sadece işlem basamaklarındaki verilerin akış yönleri farklıdır.
Kontrol transferlerde paketlerin büyüklüğü(veri paketleri) tam hız cihazlarda 8, 16, 32, 64 byte arası olabilir. Düşük hızlı cihazlarda ise maximum veri paketi büyüklüğü 8 byte'dır.Bu durumda veri paketi sadece verileri içerir, PID ve CRC alanlarını içermez.Uçnoktaların destekledikleri maximum paket büyüklüğü, uçnokta tanımlayıcısında belirtilir.Varsayılan uçnoktanın(uçnokta 0) maximum paket büyüklüğü ise cihaz ilk sisteme takıldığında Cihaz Tanımlayıcısından öğrenilir.
Kontrol transferler için ayrılan bant genişliği %10'dur.Fakat arta kalan bant genişliği varsa her bir kontrol transfer bunu eşit olarak paylaşır.
Kontrol transferler listeleme ve yapılandırma işlemleri için kullanılmalarının yanında blok bilgi transferi de yapabilir.Fakat bant genişliğinin verimli kullanılabilmesi için bu tavsiye edilmeyen bir yöntemdir.Kontrol transferler mümkün olduğunca USB istekleri için kullanılmalıdır.
Düşük hızlı cihazlar maximum 8 byte veri transferi yapabildiğinden çerçeve başına üç işlem ile sınırlıdırlar.Sadece tek bir 8 byte veri gönderimi, kontrol transferi ile yapılırsa bant genişliğinin %29'u kullanılmış olur.
Bir cihaz elsıkışma paketi yollamadığı taktirde PC toplam üç deneme daha yapar.Eğer yine yanıt alamazsa uçnokta ile haberleşme kesilir.

Kesme Transfer

Kesme transferler daha çok klavye, fare gibi cihazlarda kullanılır.PC'den cihaz'a bir istek gönderildiğinde cihaz donanımında bir kesme meydana gelir.İşte kesme transferin tam anlamı budur.Cihazda bir donanım kesmesi oluştuğunda ilgili kesme rutinine gidilir ve PC'nin isteğine karşılık gönderilmesi gereken bilgiler gönderilir veya alınır.Kesme rutinindeki kod gelen isteğin ne olduğunu algılayabilmeli, gerekli bilgileri tampona yerleştirmeli veya tampondan gerekli bilgileri okumalı,ardından uçnoktayı bir sonraki kesme transferi için hazır duruma getirip kesme rutininden çıkmalıdır.PIC18F4550 bir paket aldığında TRNIF bitini set eder.Eğer bu kesme maskelenmiş ise bu bayrağın set olmasının hemen ardından ilgili kesme rutinine gidilir.
Donanım kesmeleri herhangi bir zamanda değil, uçnokta tanımlayıcılarının bir alanındaki değer ile belirtilmiş sürelerde, PC'nin cihazı yoklaması sırasında oluşur.
Kesme transferler birden fazla IN ve OUT işlemlerinden oluşur.Şekil-28 de bu yapı gösterilmiştir ve Yığın transferlerin yapısı ile aynıdır.Şekilde de görüldüğü gibi tüm işlemler tek yönlüdür.Her işlem ya IN ya da OUT'dur.Bu yüzden iki yönde de veri alışverişi yapabilmek için her iki yöne ait ayrı transfer ve borulara ihtiyaç vardır.Tam hızlı bir cihaz için maximum paket büyüklüğü 64 byte iken düşük hızlı cihazlar için bu değer 8 byte olarak belirlenmiştir.PC bu büyüklükleri daha sonra göreceğimiz tanımlayıcılardan öğrenir.Daha öncede belirtildiği gibi donanım kesmeleri PC'nin cihazı yoklaması sırasından gerçekleşir.Bu işlemi cihaz sürücüsü gerçekleştirir ve donanımdan donanıma farklılık gösterir.Örneğin fare ve klavye gibi aygıtlar sürücü tarafından sürekli yoklanırken, bunun haricindeki başka cihazların PC'deki uygulama yazılım herhangi bir veri göndermedikçe yoklanması gerekmez.Yoklama aralığı uçnokta tanımlayıcısında belirtilir ve IN ve OUT işlemlerinin kaç ms aralıklarla yapılacağını belirler.Belirlenen bu zaman aralıklarında kesme transferinin tam olarak yapılması gerekir.Örnek vermek gerekirse biz tüm uygulamalarımızda yoklama süresini 10ms olarak belirleyeceğiz.Bu durumda 3 işlem yapılması gerekiyorsa bu üç işlem 30ms'de tamamlanacaktır.(3 x 10ms) Fakat bu durum PC tarafından idare edildiği için 4ms'de de tamamlanabilir.Çünkü bir işlem önceki işlemin bittiği zaman ölçü alınarak başlatılır.Bir boru kesme transfer için hazırlanırken uçnoktanın desteklediği tampon büyüklüğü mevcut bantgenişliği ile karşılaştırılır.İstenilen tampon büyüklüğü bantgenişliğinden büyük ise istek geri çevrilir.Bu durumda cihaz daha düşük bir tampon büyüklüğü ile istekte bulunmalıdır.Fakat kesme transferler düşük bantgenişliği ile yetindiğinden bu durumla çok sık karşılaşılmaz.



Şekil-28) Kesme Transfer IN ve OUT işlemi

Tüm transferlerde Elsıkışma paketini alamayan cihaz için PC iki kez daha denemede bulunacaktır.Bunun sonucunda yine paket alınamamışsa bu durumda uçnokta ile haberleşme kesilir.

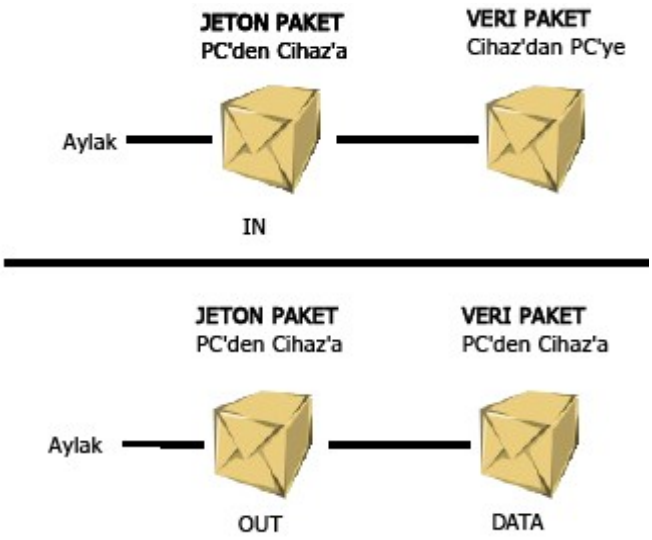
Yığın Transfer

Yığın transferler daha çok büyük miktarda verilerin bus'a gönderilmesi için kullanılan bir transfer türüdür.Bu şekilde verilerin Bus'da yığılmaya yol açmadan gönderilmesi sağlanmış olunur.Her yığın transfer aynı kesme transferdeki gibi IN ve OUT işlemlerinden oluşur ve yapısı Şekil-28'de gösterilen kesme transfer yapısı ile aynıdır.İletişimin çift taraflı yapılabilmesi için IN ve OUT olmak üzere iki ayrı boru gerekir.
Yığın transferlerde verinin gönderileceği garanti edilsede bu transfer türü için bantgenişliği tahsis edilmez.Çünkü yığın transferler diğer transferlere göre her zaman ikinci plandadır.Kontrol transferler bantgenişliğinin %10'nun kullanırken(tam hız cihazlarda) arta kalan bantgenişliği izokron ve kesme transferleri tarafından kullanılır.Bu yüzden tüm bantgenişliği kullanılıyorken, yani Bus tamamen meşgul iken yığın transferler her zaman bekletilir ve yığın transferin gerçekleşmesi çok uzun zaman alabilir.Fakat Bus meşgul değil ise yani IDLE(aylak) durumda ise bantgenişliğinin tümünü yığın transferler kullanır.Örnek olarak 19 adet 64 byte'lık bir yığın işlemi tam hız bir cihaz'da saniyede 1216 byte ile bir çerçevede tamamlanır ve bantgenişliğinin %18'i serbest kalır.
Buraya kadar incelenen transfer türlerini, geliştireceğimiz uygulamalarda yeri geldikçe kullananacağız.

Izokron Transfer

Bu transfer türünde verilerin bir akış halinde ve sabit bir hızda iletildiği kabul edilir fakat bunun muhakkak böyle olmasıda gerekmez.Ancak çoğu durumda bu şekilde kullanılır.Bu transfer türünü kullanan cihazlar daha çok audio uygulamalarıdır.Sesin bir kaynaktan alınıp digital veriye dönüştürüldükten sonra hızlı bir biçimde ve blok halinde PC'ye gönderilmesi gerekir.Bu durumda izokron transfer kullanılabileceği gibi yığın transfer'de kullanılabilir.

Fakat izokron transferler blok verinin Bus meşgul olması durumunda bile en hızlı şekilde PC'ye teslim edilmesini sağlar.Oysaki Yığın transferleri incelerken hatırlayacağınız gibi eğer Bus meşgul ise yığın transferler askıya alınır.
Izokron kelimesi senkron anlamına gelir ve her çerçevede belirli miktarda sabit veri gönderilir.Kesme transferlerin 1ms'lik (çerçeve süresi) süresi dışında tüm transfer türleri asenkron olarak gerçekleşir.Çünkü çerçeve başına belli bir sayıda byte gönderileceği kesin değildir.
İzokron transfer her çerçevede bir adet IN ve OUT işleminden oluşur.Bu yüzden her transfer sadece bir yöne doğrudur ve iki yönlü iletişim için ayrı borular gerekir.Şekil-29'da bir izokron transfer'in yapısı gösterilmiştir. Dikkat edecek olursanız işlem basamakları sadece IN ve OUT'dan oluşur ve her işlem basamağı sadece JETON ve VERI paketi içerir, ELSIKIŞMA paketi içermez.Çünkü gönderilen veriler sabit hızda ve akış şeklinde olduğu için, alınan veriler hataları ile birlikte alınır.
Diğer transfer türleri incelenirken BUS'taki bantgenişliğinin ne kadarını kullandıklarına değinilmişti.İzokron transfer çerçeve başına bir defadan, maximum 1023 byte veri gönderebilir ve alabilir.Bu durumda bantgenişliğinin %69'u bu transfer tarafından kullanılır.Bu yüzden bir transfer borusu izokron olarak yapılandırılmadan önce istenilen tampon büyüklüğü (burada 1023 byte'lık) PC tarafından boş bantgenişliği ile kıyaslanarak, bantgenişliğinin mevcut olup olmadığı belirlenir.Eğer bantgenişliği mevcut değil ise cihaz daha küçük bir tampon boyutu ile tekrar istekte bulunmalıdır.Burada ve diğer transfer türleri açıklanırken konusu geçen "cihaz tekrar istekte bulunmalıdır" sözcüğü, cihazın mevcut konfigrasyonla geri çevrilmesinin ardından başka alternatifleri varsa, cihaz sürücüsü'nün bu ayarlar ile tekrar istekte bulunması olarak algılanmalıdır.Yani bu işlem sürücü tarafından yürütülür.



Şekil-29) Izokron transfer

USB Cihazların Sisteme Tanıtılması

Daha önceki ünitelerde USB arabiriminin diğer protokollere göre çok daha zor ve karmaşık olduğunu vurgulamıştık.RS232 veya paralel portu kullanan bir cihaz sisteme takıldığında haberleşmeye hazırdır.Eğer uygulama yazılımı ve cihaz yazılımı aynı konfigrasyona sahip ise (baud hızı, eşlik biti...) hemen haberleşmeye başlanabilir.Fakat USB cihazlarda durum çok daha farklıdır.Bir USB cihaz sisteme takıldığı zaman hemen haberleşmeye girişilemez.Cihaz ilk olarak Windows tarafından listeleme işlemine sokulur ve incelenmeye başlanır.Listeleme sırasından Windows cihaz'a daha sonra göreceğimiz bazı isteklerde bulunur.Cihaz bu istekleri algılayabilecek, desteklediği istekleri gerektiği gibi yanıtlayacak ve desteklemediklerini ise uygun bir cevap ile bildirecek yeteneğe sahip olmalıdır.Bu durumda yonga kodu'na çok görev düşmektedir.
Ayrıca cihazlar her isteği desteklemek zorunda olmadıkları gibi bazılarını desteklemek zorundadırlar. Desteklenmeyen istekleri ise STALL ile uygun bir şekilde geri çevirmelidirler.İstekler daha sonradan da inceleneceği gibi PC'nin, cihazın yeteneklerini öğrenebilmesi(transfer türü ve uçnoktalar vb..) için gereklidir. Listeleme işleminde bir sorun çıkmaz ise PC cihaz için bir adres tayin eder ve artık veri alışverişi bu adres üzerinden gerçekleşir.

Her cihaz listeleme işleminin yapılabilmesi için varsayılan kontrol uçnoktası olarak Uçnokta 0'ı desteklemek zorundadır.Aynı zamanda bu uçnokta hem IN hemde OUT transferi desteklemek durumundadır.Listeleme işlemi kontrol transfer'ine başvurularak yapılır ve bu uçnokta üzerinden hem veri alınır hemde veri gönderilir. Listeleme işlemi kontrol transferi kullanılarak yapıldığından bu transfer türü daha sonra daha detaylı bir şekilde incelenecektir.Listeleme işlemi sorunsuz tamamlandıysa cihaz Aygıt Yöneticisine yerleşir ve veri alışverişine hazır hale gelir.

Listeleme İşlemi ve Adımları

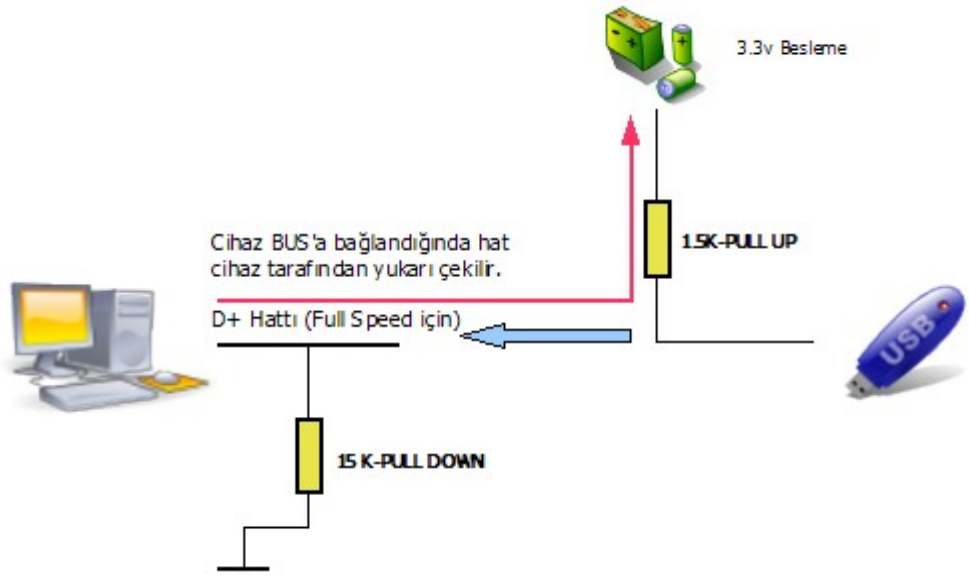
Şimdi bir USB cihaz sisteme takıldığı zaman gerçekleşen olayları adım adım inceleyelim.Burada anlatılanlar ilerleyen bölümlerde cihaz tasarımı bittiğinde ve sisteme ilk takıldığında Device Monitoring Studio aracı ile daha detaylı incelenektir.
Bir cihaz herhangi bir anda altı durumdan sadece birine sahip olabilir.Bu durumlar aşağıda listelenmiştir;

- DETACHED : Cihaz BUS'a bağlı değil
- ATTACHED : Cihaz BUS'a bağlı
- DEFAULT : Cihaz varsayılan başlangıç durumunda
- POWERED : Cihaz açık
- ADDRESS : Cihaz'a bir adres atanmış
- CONFIGURED : Cihaz konfigrasyonu tamamlanmış ve hazır

Her cihaz listeleme aşamasında bu altı durumdan sırası ile geçer ve son aşamada cihaz'a bir konfigrasyon değeri atanarak veri alışverişi için hazır hale getirilir.

1- PC'nin HUB'ının USB Cihazı Algılaması

PC'nin USB portuna takılan bir cihaz'ın algılanması, sinyal hattındaki voltaj değişimi sayesinde olur.Çünkü PC'nin Hub'ı USB portlarının sinyal hatlarındaki voltaj seviyesini denetler.
PC'deki her USB portu'nun D+ ve D- hatlarında 15 KiloOhm'luk aşağı çekmeli dirençler vardır.(Pull-Down) Bir USB cihaz'ın ise eğer cihaz tam hızlıysa D+ hattında, eğer cihaz düşük hız ise D- hattında 1.5 KiloOhm-luk yukarı çekmeli direnç bulunmalıdır.PIC18F4550'de bu dirençler dahili olarak mevcuttur ve hangi hattın 1.5K ile yukarı çekileceği yazılımsal olarak kontrol edilir.
Cihaz USB portuna takıldığında, cihazın D+ veya D- hatlarında bulunan 1.5K'lık yukarı çekmeli direnç, PC'nin portunda bulunan 15K'lık aşağı çekmeli hattı beslemeye çeker.Böylece PC'nin Hub'ı sisteme yeni bir cihaz takıldığını algılar.Şekil-30'da hattın nasıl beslemeye çekildiğini gösteren şekil verilmiştir.



Şekil-30) USB cihaz'ın algılanması

PC'nin HUB'ında bu olayların meydana geldiğinin bilinmesi için bir Kesme IN borusu vardır.Olay ya USB portlarının birinde ya da HUB'da meydana gelir.Eğer olay portlardan birinde meydana gelirse hangi potta gerçekleştiğini işaret eden bildirim gönderilir.
Bu işlemlerin ardından PC detaylar için HUB'ına Get_Port_Status isteğini gönderir.Cihaz bu durumda ATTACHED durumundadır.

2- Cihaz'ın PC tarafından Resetlenmesi

PC'nin HUB'ı yeni bir cihazın takıldığını algıladıktan sonra HUB'a PC tarafından Set_Port_Feature isteği gönderilerek HUB'dan cihazı reset etmesi istenir.Normalde D+ ve D- hatlarının lojik seviyeleri birbirinin zıttıdır. Fakat Reset isteği gönderildiğinde D+ ve D- hatları HUB tarafından 10ms boyunca düşük seviyede tutulur. HUB reseti bıraktığında cihaz DEFAULT durumundadır.Bu durumda cihaz uçnokta 0 üzerinden ve adres değeri 0 olmak üzere haberleşmeye hazırdır.Cihazın bu durumda BUS'dan 100ma'den daha fazla akım çekemez.Daha sonrada inceleneceği gibi cihaz ancak konfigrasyon künyesi incelendikten ve max akım değeri onaylandıktan sonra BUS'dan 500ma kadar akım çekebilir.Fakat DEFAULT durumda BUS cihaza sadece 100ma akım sağlar. PC ise cihazın reset durumdan çıkıp çıkmadığını yani HUB'ın reset'i tamamlayıp tamamlamadığını HUB'a Get_Port_Status isteğini göndererek öğrenir.Gönderilen cevaptaki bir bit cihazın hala resette ya da resetten çıktığını gösterir.

3- Cihaz Hızının HUB Tarafından Algılanması

Reset işleminin hemen ardından HUB sinyal hatlarındaki voltaj seviyelerini denetler.D+ ve D- hatlarından hangisinin yüksek seviyede olduğunu inceler ve cihazın full-speed ya da low-speed olduğuna karar verir.

4- Uçnokta 0'ın Maximum Paket Büyüklüğünün Öğrenilmesi

Tüm bu hazırlık işlemleri tamamlandıktan sonra listelemenin başlaması gerekir.Listeleme işlemi daha öncede söylendiği gibi PC'nin cihaz'a standart ve yanıtlaması zorunlu istekler göndermesi ve yanıtları değerlendirmesi işlemidir.Bu yüzden yapılacak ilk şey listeleme işleminin kontrol transfer yolu ile yürütüleceği uçnokta 0'ın maximum paket büyüklüğünün öğrenilmesidir.Bu işlem için PC cihaz'a Get_Descriptor isteği gönderir.Bu istek " Tanımlayıcı Gönder" isteğidir ve JETON paketi PaketID'si SETUP, VERI paketi içeriği ise hangi tanımlayıcının istendiği ve diğer ek bilgiler ile doludur.Bu anlatılanlar ilerleyen bölümlerde detaylı bir şekilde anlatılmıştır. Uçnokta 0'ın maximum paket büyüklüğü Cihaz tanımlayıcısının sekizinci byte'ındadır.Bu yüzden PC ilk sekiz byte'ı okuyacak ve hemen ardından STATUS basamağını başlatarak transferi sonlandıracaktır.Transferin sonlanmasının hemen ardından artık PC uçnokta 0'ın maximum paket büyüklüğünü biliyordur.PC HUB'a cihazı tekrar reset etmesini söyler.
Bu istekler adres bilgisi 0 olan cihaz'ın uçnokta 0'ına gönderilir.Sistemde adres bilgisi 0 olan sadece bir tek cihaz olabileceğinden bu isteklerden diğer cihazlar etkilenmez.Çünkü sadece yeni cihaz adresi 0 olabilir ve belli bir anda sadece bir tek cihaz listelemeye alınır.Artık cihaz adres bekliyordur.

5- Cihaz'a Yeni Adres Atanması

Her cihaz ortak veriyolunu kullandığından kendine ait bir adres'i vardır.Bu adres PC tarafından listeleme sırasında atanır ve cihaz sistemden sökülene kadar aynı kalır.Bir sonraki listelemede bu adresin aynı olması gerekmez.Cihaz adresleri 0 ile 127 arasında olabilir.Artık cihaz ADDRESS durumundadır.

6- Cihaz'ın Özelliklerinin Öğrenilmesi

Cihaz'a adres atanmasının hemen ardından PC cihazdan Cihaz Tanımlayıcısının tamamını istediğini içeren Get_Descriptor isteğini gönderecektir.Cihaz Tanımlayıcısı maximum konfigrasyon adedi, uçnokta 0'ın maximum paket büyüklüğü VendorID, ProductID gibi bilgiler içerir.
Bu tanımlayıcının okunmasının hemen ardından PC, cihaz tanımlayıcısında belirtilen konfigrasyon tanımlayıcısının ilk dokuz byte'ını istediğini bildiren bir Get_Descriptor isteği gönderir.Daha sonra da inceleneceği gibi konfigrasyon tanımlayıcısının bir alanında(wTotalLength) bu konfigrasyona ek olarak okunacak uçnokta, arabirim ve sınıf gibi tanımlayıcıların toplam uzunluğu bulunur.İlk konfigrasyon tanımlayıcısının dokuz byte'ı okunduktan sonra(zaten bu tanımlayıcı boyutu dokuz byte'dır) diğer tanımlayıcıların toplam uzunluğu öğrenilmiştir.PC bu defa konfigrasyon tanımlayıcısını yeniden istediğini belirten Get_Descriptor isteğini gönderir.

Bu defa 255 byte'a kadar toplam uzunluk istendiğinden konfigurasyon tanımlayıcısının ardından arayüz tanımlayıcısı ve uçnokta tanımlayıcılarıda gönderilir.PC gönderilen bu bilgileri haberleşme esnasında ve sürücü seçiminde kullanacaktır.

7- Cihaz Sürücüsü'nün Yüklenmesi

PC cihaz'dan okuduğu tanımlayıcılardaki bilgiler doğrultusunda cihaz için en uygun sürücüyü aramaya başlar.Bu işlem INF dosyaları yardımıyla gerçekleşir.Tanımlayıcılardan okunan VendorID, ProductID, Sürüm Numarası ve cihaz'ın uyumlu olduğu sınıf bilgileri INF dosyaları içerisinde aranır.Bu bilgiler ile uyuşan INF dosyası bulunduğunda, bu INF dosyasının belirttiği yerdeki sürücü belleğe yüklenir.
Sürücünün yüklenmesinin hemen ardından PC cihazdan kendisi ile ilişkili sınıf tanımlayıcılarını tekrar gönderme-
sini isteyebilir.

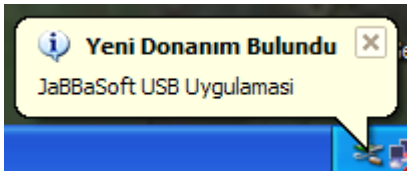
8- PC Tarafından Uygun Konfigrasyonun Set Edilmesi

Sürücü'nün yüklenmesinin hemen ardından yapacağı ilk iş cihaz'da mevcut bulunan ve istenen konfigrasyonu bir Set_Configuration isteği ile set etmesidir.Artık cihaz'ın arayüzü devreye girmiştir ve CONFIGURED durumundadır.

Cihaz daha önce anlatılan altı duruma ek olarak birde SUSPEND durumunda olabilir.Bu durum cihaz BUS'da 3 ms faliyet görmediği durumda meydana gelir ve cihaz'ın bu modda BUS'dan çektiği akım çok düşüktür.Fakat haberleşmeye her zaman hazırdır.
Yukarıda anlatılan listeleme adımları, tasarladığımız devreyi test ederken Device Monitoring Studio programı ile daha detaylı incelenecek ve daha rahat anlaşılacaktır.

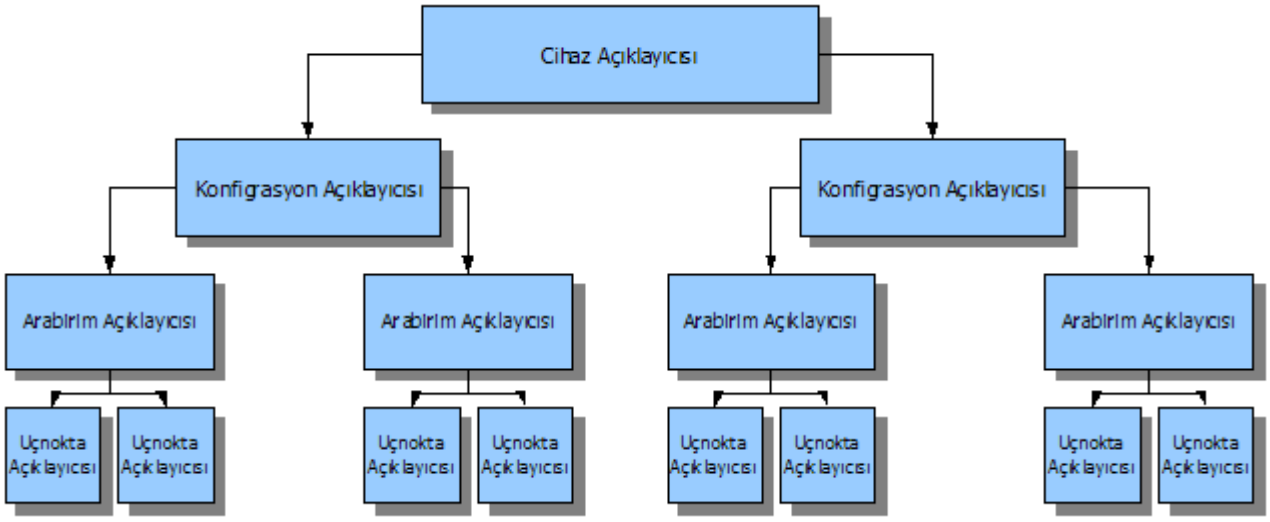
Tanımlayıcılar ve İçerikleri

Tanımlayıcılar transfer türü, ürün ve üretici kodları, seri numarası, uçnokta özellikleri, güç seçenekleri ve daha birçok bilgiyi içeren, belirli bir formatta sıralanmış veri bloklarıdır.
PC bu veri bloklarını gönderdiği istekler doğrultusunda ister ve cihaz hakkında bilgi toplamış olur.Örneğin daha sonra tasarlayacağımız devre'de PIC18F4550 için tanımlanmış String Tanımlayıcısı içeriği "JabbaSoft USB Uygulamasi" şeklindedir.PC, Get_Descriptor isteğini gönderirse ve bu isteğin içeriği String_Descriptor, yani String Tanımlayıcısı ise cihaz "String Tanımlayıcısını" yani yukarıdaki karakter katarını gönderir.Böylece siz mesaj balonunda aşağıdaki gibi bir mesaj alırsınız.
Şimdi hepinizin "Demek ki flash belleğimi takdığımda bu balonda çıkan mesajlar meğer üretici tarafından tanımlanmış karakter katarlarıymış" dediğini duyar gibiyim ya da öyle demenizi umuyorum. :)



Şekil-31) String Tanımlayıcısı'nın Okunması

Bu veri bloklarının nasıl tanımlandığını PIC Firmware'ını incelerken ve şimdi Tanımlayıcı türleri incelerken daha iyi anlayacaksınız.Tanımlayıcıların hiyerarşik bir yapısı vardır.Şekil-32'de bu yapı gösterilmektedir.
Şekilde de görüldüğü gibi Cihaz Tanımlayıcısı sadece cihazın tümüne ilişkin özel bilgileri içerir.Konfigrasyon Tanımlayıcısı ise bir veya birden fazla olabilir ve adedi Cihaz Tanımlayıcısında belirtilmiştir.Her Konfigrasyon Tanımlayıcısı arayüz, uçnokta ve sınıf tanımlayıcılarının toplam uzunluğunu taşır.Aynı zamanda maximum arabirim sayısını içerir.Bu tanımlayıcının hemen ardından biraz önce de bahsedildiği gibi, Konfigrasyon Tanımlayıcısında adedi tanımlı Arabirim Tanımlayıcıları gelir.Her Arabirim Tanımlayıcısı desteklenen uçnokta adedini ve diğer ek bilgileri içerir.Aynı zamanda cihazın uyumlu olduğu sınıf bilgisini içerir.Bu tanımlayıcının hemen ardından Sınıf ve Uçnokta Tanımlayıcıları yer alır.Uçnokta Tanımlayıcılarının adedi Arabirim Tanımlayıcısında belirtilen adet kadar olmalıdır.Bu tanımlayıcı tanımlanan uçnokta'nın hangi transfer türünü desteklediğini, maximum paket boyutunu ve yoklanma aralığı gibi bilgiler içerir.Şimdi bu tanımlayıcıları daha detaylı inceleyeceğiz.



Şekil-32) Tanımlayıcıların Hiyerarşik Yapısı

Her tanımlayıcı kendi tipini belirten bir değer taşır.Aşağıdaki tabloda USB Spesifikasyonunda tanımlı tanımlayıcılar gösterilmiştir.Burada verilen tanımlayıcılar önceden belirlenmiş sabit değer'leri ile ayırt edilirler.7.bit daima 0'dır.6. ve 5. bitler tanımlayıcının tipini belirtir.Bu bitler 00 Standart, 01 Sınıf, 10 Üretici ve 11 Rezerve değerlerinden herhangi birini alabilir.4-0 arası bitler ise tanımlayıcıyı tanımlar.

Tip	Değer	Açıklayıcı
Standart	0x01	Aygıt
	0x02	Konfigrasyon
	0x03	String
	0x04	Arabirim
	0x05	Uçnokta
	0x06	Cihaz Niteleyeci
	0x07	Diğer Hız Konfigrasyonu
	0x08	Arabirim Gücü
Sınıf	0x21	HID
	0x29	HUB
HID'a Özgü	0x22	Rapor
	0x23	Fiziksel

Tablo-2) Tanımlayıcı Tipleri

Standart bölümündeki tanımlayıcıların değerlerine dikkat edin.7.bit 0, 6 ve 5.bitler 00 yani Standart, diğer 4-0 arası bitler ise tanımlayıcı tipini göstermektedir.(1,2,3..)
Sınıf bölümündeki tanımlayıcılardan HID'ı örnek alırsak 7.bit yine 0, 6 ve 5.bitler 01 yani Sınıf, diğer 4-0 arası bitlerin ise tanımlayıcı tipini gösterdiğini görürüz.

Aygıt Tanımlayıcısı / *Device Descriptor*

USB Cihaz sisteme ilk bağlandığında PC'nin cihazdan istediği ilk tanımlayıcı Aygıt Tanımlayıcısıdır.PC bu tanımlayıcıyı diğer tanımlayıcıları aldığı gibi Get_Descriptor isteği ile alır.Daha önceden, listelemeye başlamak için PC'nin uçnokta 0'ın maximum paket büyüklüğünü bu tanımlayıcıdan okuduğunu hatırlayın.
Cihaz Tanımlayıcısı cihaz ile ilgili temel bilgileri ve diğer ihtiyaç duyulan bilgileri içerir.Tablo-3'de bu tanımlayıcının alanları gösterilmiştir.

OffSet	Alan	Boyut	Açıklama
0	bLength	1	Bu açıklayıcının byte olarak uzunluğu
1	bDescriptorType	1	Açıklayıcı tipi.Bu alan sabit DEVICE(0x01)'dır.
2	bcdUSB	2	USB Spesifikasyon sürüm numarası(BCD)
4	bDeviceClass	1	Cihaz'ın uyumluğu olduğu sınıf kodu
5	bDeviceSubClass	1	Alt Sınıf kodu
6	bDeviceProtocol	1	Protokol kodu
7	bMaxPacketSize	1	Uçnokta 0'ın maximum paket büyüklüğü
8	idVendor	2	Üretici kimliği
10	idProduct	2	Ürün kimliği
12	bcdDevice	2	Cihaz sürüm numarası(BCD)
14	iManufacturer	1	Üretici String Açıklayıcı index'i
15	iProduct	1	Ürün String Açıklayıcı index'i
16	iSerialNumber	1	Seri numarası String Açıklayıcı index'i
17	iNumConfigurations	1	Destelenen konfigrasyon adedi

Tablo-3) Aygıt Tanımlayıcısı

Bu tanımlayıcının toplam 18 alanı vardır.Her alanın boyutu byte cinsinden ifade edilir.Diğer tanımlayıcılarda olduğu gibi bu tanımlayıcıda uzunluğunu ve tipini gösteren bilgiler ile başlar.PC gönderilen tanımlayıcının tipini bu bilgiler sayesinden tespit eder.Şimdi bu tanımlayıcının alanlarını inceleyelim.

bLength
Bu alan Cihaz Tanımlayıcısının toplam boyutu tutar.Cihaz Tanımlayıcısının toplam boyutu OffSet değerinin sonundan da anlaşılacağı gibi 18 byte'dır.

bDescriptorType
Bu alan Tablo-2'de gösterilen DEVICE(0x01) değerini alır.PC bu tanımlayıcının Cihaz Tanımlayıcısı olduğunu bu değer ile algılar.

bcdUSB
USB Spesifikasyon sürüm numarasını tutar.Bu alan BCD formatındadır ve örnek verecek olursak version 1.1 için 0110, version 1.0 için 0100 değerlerini alabilir.

bDeviceClass
Bu alan USB için tanımlanmış sınıflardan birini içerebilir.USB için tanımlanmış sınıflar 0x01 ile 0xFE arasındadır, 0xFF değeri ise sınıfın üretici tarafından belirlendiğini gösterir.HID sınıfı ve diğer bazı cihazların sınıf kodu bu alanda değil arabirim tanımlayıcısında tanımlanır.Bu yüzden bu alan 0x00 olarak geçilir.

bDeviceSubClass
Bu alan bDeviceClass'ın alt sınıf değerini tutar.Eğer bDeviceClass 0x01 – 0xFE değerinde ise bu alanıda tanımlı bir değeri olmalıdır.Eğer bDeviceClass 0x00 ise bu alanda 0x00 olmalıdır.

bDeviceProtocol
Bu alan protokol kodunu tutar.Protokol koduna arabirim tanımlayıcısı incelenirken değinilecektir.

bMaxPacketSize
Bu alan Uçnokta 0'ın maximum paket büyüklüğünü tutar.Daha önceden listeleme aşamasında PC'nin ilk olarak Cihaz Tanımlayıcısından bu alanı okuduğunu hatırlayın.

idVendor
Bu alan üretici kimlik bilgisini tutar ve her cihaz tanımlayıcında bulunmak zorundadır.Daha öncede anlatıldığı gibi PC idVendor ve idProduct alanlarından okuduğu değerleri INF dosyalarında arar ve uygun sürücüyü yükler.Bu alandaki değer aslında USBIF(Usb Implementers Forum) tarafından sadece üyelere ve aidat veren kurumlara verilir.

idProduct
Bu alandaki değer üretici tarafından ürüne tahsis edilmiş ürün kimliğini içerir.Her cihaz künyesinde bulunmak zorundadır.

bcdDevice
Bu alan cihazın sürüm numarasını içerir.BCD formatındadır ve cihazın üreticisi tarafından verilir.Tanımlanması zorunlu olmasada sürücü seçiminde kullanılabilir.

iManufacturer
Bu alan üretici string index'ini tutar.PC bu alanı okuduğunda üretici string'inin index'ini öğrenmiş olur ve Get_Descriptor ile String Tanımlayıcısını isterken önceden öğrendiği bu index değerini kullanarak Üretici string'ini istediğini belirtir.Kullanılmayacaksa 0 geçilmelidir.

iProduct
Bu alan ürün string'inin index'ini tutar.İşlemler iManufacturer alanı açıklanırken anlatıldığı gibidir.Kullanılmaya-caksa 0 geçilmelidir.

iSerialNumber
Bu alan cihazın seri numarası string'inin index'ini tutar.Kullanılmayacaksa 0 geçilebilir.Fakat aynı tip birkaç cihaz BUS'a bağlanmış ise, PC cihazları seri numaralarından ayırt ederek tespit eder.Aynı zamanda seri numarası string'i tanımlanmış bir cihaz farklı portlara takıldığında PC bu cihazın daha önceden kurulu olup olmadığını tespit eder ve yeniden sürücü yükleme girişiminde bulunmaz.

bNumConfigurations
Bu alan cihazın desteklediği konfigrasyon adedini içerir.

Konfigrasyon Tanımlayıcısı / *Configuration Descriptor*

Şekil-32'de görüldüğü gibi tanımlayıcıların hiyararşik bir yapısı vardır ve bu sıralamanın en üstünden Cihaz Tanımlayıcısı bulunur.PC Cihaz Tanımlayıcısını okuduktan hemen sonra Konfigrasyon, Arabirim, varsa Sınıf tanımlayıcıları ve Açnokta Tanımlayıcılarını isteyecektir.Her istek cihazın daha küçük bir elemanına ilişkindir. Konfigrasyon Tanımlayıcısı cihazın özellikleri ve kapasitesi hakkında bilgiler içerir.Güç kullanımı ve uzaktan uyandırma, arabirimlerin sayısı gibi özellikler örnek olarak verilebilir.Arabirim Tanımlayıcıları ve bir veya daha fazla Uçnokta Tanımlayıcıları Konfigrasyon Tanımlayıcısının alt elemanlarıdır.PC Konfigrasyon Tanımlayıcısını isterken toplam uzunluğu isterse tüm bu tanımlayıcılarında cihaz tarafından gönderilmesi gerekir.Tablo-4'de Konfigurasyon Tanımlayıcısının alanları gösterilmiştir.

OffSet	Alan	Boyut	Açıklama
0	bLength	1	Bu açıklayıcının byte olarak uzunluğu
1	bDescriptorType	1	Açıklayıcı tipi.Bu alan sabit CONFIGURATION(0x02)dır
2	wTotalLength	2	Bu konfigrasyon ve alt açıklayıcıların toplam boyutu
4	bNumInterface	1	Desteklenen arabirim adedi
5	bConfigurationValue	1	Set_Configuration ve Get_Configuration için değer
6	iConfiguration	1	Konfigrasyon String Açıklayıcı index'i
7	bmAttributes	1	Güç ve uzaktan uyandırma ayarları
8	MaxPower	1	BUS güç ayarı

Tablo-4) Konfigrasyon tanımlayıcısı

Bu tanımlayıcının toplam uzunluğu 9 byte'dır.Konfigrasyon tanımlayıcısıda diğer tanımlayıcılar gibi Get_Descriptor isteği ile okunur.

bLength
Bu alan Konfigrasyon Tanımlayıcısının toplam boyutunu tutar ve sabit 9 byte'dır.

bDescriptorType
Bu alan Tablo-2'de gösterilen CONFIGURATION(0x02) değerini alır.PC bu tanımlayıcının Konfigrasyon Tanımlayıcısı olduğunu bu değer ile algılar.

wTotalLength
Bu alan Konfigrasyon Tanımlayıcısıda dahil olmak üzere alt açıklayıların toplam uzunluğunu tutar.Alt tanımlayıcılar Arabirim, Sınıf ve Uçnokta tanımlayıcılarıdır.Bu alana tüm bu tanımlayıcıların toplam uzunluğu verilmelidir.

bNumInterface
Bu alan desteklenen arabirim sayısını tutar.Buradaki adet kadar Arabirim Tanımlayıcısı tanımlanmalıdır.

bConfigurationValue
Bu alan Get_Configuration ve Set_Configuration için değer tanımını içerir.Örneğin bu alan içeriği 0x01 ise PC gerekli tanımlayıcıları okuduktan ve cihaz'ı uygun bulduktan sonra Set_Configuration ile hazırdaki konfigrasyonu set eder.İşte Set_Configuration isteğindeki wValue alanı(daha sonra incelenecektir) bConfigurationValue alanında tanımlı değeri içerir.Get_Configuration ile aktif konfigrasyon istendiğinde ise Set_Configuration ile set edilmiş konfigrasyon bilgisi gönderilir.

iConfiguration
Bu alan Konfigrasyon string'inin index'ini tutar.Kullanılmayacaksa 0 geçilebilir.

bmAttributes
Bu alanın boyutu Tablo-4'den de görüleceği gibi 1 byte'dır.7.bit daima 1 olmadır.Eğer cihaz'ın kendine ait besleme kaynağı varsa yani cihaz kendinden beslemeli ise 6.bit 1 aksi halde 0 olmadır.Cihaz kendinden beslemeli ise bu bitin 1 yapılması BUS'dan güç alınmayacağı anlamına gelir.Bu durumda PC sadece listeleme aşamasında cihaza 100ma akım sağlayacak sonra kesecektir.Cihaz BUS'dan beslemeli ise bu bitin 0 yapılması, cihazın listeleme sonrası BUS'dan 500ma kadar akım çekebileceği anlamına gelir.
Eğer cihaz uzaktan uyandırmayı (RESUME) destekliyorsa 5.bit 1 yapılabilir.Bu durumda suspend(askı)konu-mundaki bir cihaz PC'ye haberleşme isteğini bildirebilir.4-0 arası bitler ise kullanımaz, 0 olarak kalmalıdır.

MaxPower
Bu alan cihazın gereksinim duyduğu BUS akımını belirtir.MaxPower alanına gerekli olan BUS akım değerinin yarısı verilir.Örneğin cihaz 100ma'e ihtiyaç duyuyorsa bu alana 50 değeri atanır.Bu yarı değerin saklanması daha yüksek değerlerin ifadesinde bir byte kazandırmış olur.Normalde bu alanın boyutu bir byte'dır.Cihaz BUS'dan maximum 500ma kadar akım çekebilir.Bu alanın boyutu bir byte olduğunda 500 değerini ifade etmenin bir yolu yoktur.(1 byte 255'e kadar değer alır)Bu yüzden 500 değeri limit değerdir ve 250 gibi bir değer ile ifade edilir.PC bunu 500ma olarak değerlendirecektir.İzin verilen akım miktarı ise miliamper ile sınırlıdır.

Arabirim Tanımlayıcısı / *Interface Descriptor*
Arabirim tanımlayıcısı tüm USB cihazı tanımlamak için kullanılabileceği gibi aslında bir veya birden çok uçnoktayı gruplar.Cihazın desteklediği sınıf bilgisi ve protokol kodu gibi bilgilerin hepsi bu alanda yer alır.Tablo-5'de bu tanımlayıcının alanları gösterilmiştir.

OffSet	Alan	Boyut	Açıklama
0	bLength	1	Bu açıklayıcının byte olarak uzunluğu
1	bDescriptorType	1	Açıklayıcı tipi.Bu alan sabit INTERFACE(0x04)dir
2	bInterfaceNumber	1	Bu arabirimi belirleyen sayı
3	bAlternateSetting	1	Alternatif ayar değeri
4	bNumEndPoints	1	Uçnokta 0 haricinde desteklenen uçnokta adedi
5	bInterfaceClass	1	Arabirimin uyumlu olduğu sınıf
6	bInterfaceSubClass	1	Arabirimin uyumlu olduğu alt sınıf
7	bInterfaceProtocol	1	Protokol Kodu
8	iInterface	1	Arabirim String Açıklayıcı index'i

Tablo-5) Arabirim tanımlayıcısı

bLength
Bu alan Arabirim Tanımlayıcısının toplam boyutunu tutar ve sabit 9 byte'dır.

bDescriptorType
Bu alan Tablo-2'de gösterilen INTERFACE(0x04) değerini alır.PC bu tanımlayıcının Arabirim Tanımlayıcısı olduğunu bu değer ile algılar.

bInterfaceNumber
Composide bir cihazın birden çok arabirimi olabileceğini Şekil-32'deki hiyerarşik yapıda görmüştük.

Bu yüzden her arabirim'in diğerlerinden ayırt edilebilmesi için bu alanında tanımlı benzersiz bir numarası olması gerekir.İlk değer 0'dan başlar.

bAlternateSetting
Konfigrasyon Tanımlayıcındaki *bConfigurationValue* alanındaki gibi bir anlamı vardır.PC hazırdaki bir arabirimi Set_Interface ederken burada tanımlanan değeri kullanır.Get_Interface ile de önceden tanımladığı değeri ister.İlk değer 0'dan başlar.

bNumEndPoints
Bu arabirimin uçnokta 0 hariç(default kontrol uçnoktası)desteklediği uçnokta adedini gösterir.Eğer cihaz listeleme ve yapılandırma işlemleri hariç, veri iletişimi içinde kontrol transferini kullanacaksa ve başka uçnokta tanımlamamışsa bu alan 0 değerini alabilir.

bInterfaceClass
Bu alan Cihaz Tanımlayıcısında bulunan DeviceClass'a benzer.Çoğu cihaz hangi sınıfa ait olduğunu bu alanda belirtir.Örneğin HID cihazlar sınıf kodunu Cihaz Tanımlayıcısının DeviceClass alanında değil bu alanda tanımlar ve bu alan 0x03 değerini alır.(HID kodu)

bInterfaceSubClass
Bu alan Cihaz Tanımlayıcısında bulunan DeviceSubClass'a benzer.Çoğu cihaz alt sınıf kodunu yine bu alanda belirtir.*bInterfaceClass* alanı HID olan bir arabirim için bu alanın alabileceği değerler şunlardır;

SubClass Kod	Açıklama
0x00	Alt sınıf tanımlı değil
0x01	Boot arabirimli alt sınıf
0x02 – 0xFF	Reserve

Bu alanın değeri 0x01 ise cihaz boot arabirimine destek veriyor demektir.Boot arabirimine destek veren bir cihaz PC'nin HID sürücüleri yüklü değilken de kullanılabilir.Örneğin bir cihaz boot arabirimine destek veriyorsa DOS modunda açılan PC'de de çalışabilecektir.

bInterfaceProtocol
Bu alan Cihaz Tanımlayıcısında bulunan bDeviceProtocol'a benzer ve arabirimin hangi protokolü desteklediğini belirtir.Eğer bInterfaceSubClass alanı 0 ise bu alanda 0 olmalıdır.Yine HID sınıfını örnek verecek olursak bu alan sadece bInterfaceSubClass alanı boot arabirimini destekliyorsa tanımlanabilir aksi halde 0 değerini alır. Eğer arabirim boot arabirimini destekliyorsa bInterfaceProtocol aşağıdaki değerlerden birini alabilir;

Protokol Kod	Açıklama
0x00	Protokol tanımlı değil
0x01	Klavye
0x02	Fare
0x03 – 0xFF	Reserve

iInterface
Bu alan Arabirim String'inin index'ini tutar.

Uçnokta Tanımlayıcısı / *EndPoint Descriptor*
Uçnokta tanımlayıcıları destekledikleri transfer tipini, maximum paket büyüklüklerini ve IN veya OUT olduklarını belirten bilgileri içerir.Arabirim Tanımlayıcısında belirtilen her uçnokta'nın kendine ait bir tanımlayıcısı olmak zorundadır.Örneğin Arabirim Tanımlayıcısının bNumEndPoints alanı 0x02 ise iki adet Uçnokta Tanımlayıcısı tanımlanacak anlamına gelir.Tablo 6-)'da Uçnokta Tanımlayıcısının alanları gösterilmiştir.

OffSet	Alan	Boyut	Açıklama
0	bLength	1	Bu açıklayıcının byte olarak uzunluğu
1	bDescriptorType	1	Açıklayıcı tipi.Bu alan sabit ENDPOINT(0x05)
2	bEndPointAddress	1	Uçnokta adresi ve yönü
3	bmAttributes	1	Desteklenen transfer tipi
4	wMaxPacketSize	2	Uçnoktanın desteklediği maximum paket boyu
6	bInterval	1	Max gecikme, yoklama ve NAK adedi

bLength
Bu alan Uçnokta Tanımlayıcısının toplam boyutunu tutar ve sabit 6 byte'dır.

bDescriptorType
Bu alan Tablo-2'de gösterilen ENDPOINT(0x05) değerini alır.PC bu tanımlayıcının Uçnokta Tanımlayıcısı olduğunu bu değer ile algılar.

bEndPointAddress
Bu alan tanımlanan uçnoktanın IN veya OUT olduğunun bilgisini ve uçnokta numarasını tutar.0-3 arası bitler uçnokta numarasını belirtir.Düşük hızlı bir cihaz maximum 3 uçnokta tanımlayabilirken, tam hızlı bir cihaz 16 adet uçnokta bulundurabilir.İşte bu alandaki 0-3 arası bitler maximum uçnokta adedini tanımlayabilir.(0-3 arası bitler 2^n den 16 adet uçnokta tanımlar)7.bit yön bitidir.Bu bitin 0 olması uçnoktanın OUT olduğunu,1 olması ise IN olduğunu belirtir.4-6 arası bitler kullanılmaz.

bmAttributes
Bu alan uçnoktanın desteklediği transfer tipini ve sürüme göre diğer bilgileri saklar.0 ve 1 numaralı bitler transfer tipini ifade etmek için kullanılır ve aşağıdaki değerlerden birini alabilir, varsayılan kontrol uçnoktasının transfer tipi her zaman Kontrol'dür;

Değer	Transfer Tipi
0x00	Kontrol
0x01	Izokron
0x02	Yığın
0x03	Kesme

USB 1.1 sürümünde 2 ile 7 arası toplam beş bit ayrılmıştır(Reserve) fakat USB 2.0'da 2'den 5'e kadar olan dört bit kullanılmaktadır.Bunlardan 2 ve 3 numaralı bitler senkronizasyon tipini ifade ederler ve aşağıdaki değerleri alabilirler;

Değer	Senkronizasyon Tipi
0x00	Senronizasyon Yok
0x01	Asenkron
0x02	Adaptif
0x03	Senkron

Diğer 4. ve 5. bitler ise kullanım tipini gösterir ve aşağıdaki değerleri alabilirler;

Değer	Kullanım Tipi
0x00	Veri uçnoktası
0x01	Geri besleme uçnoktası
0x02	Zımni geri besleme uçnokta
0x03	Ayrılmış(Rezerve)

6 ve 7.bitler daima 0'dır, Izokron olmayan uçnoktalarda da 2 ile 5 arası bitler 0 olmalıdır.

wMaxPacketSize
Bu alan uçnoktanın her transferde gönderip alabileceği maximum paket büyüklüğünü tanımlar.
USB 2.0 altındaki sürümlerde 0'dan 10'a kadar olan bitler paket büyüklüğünü gösterir.Bu değer 0-1023 arası olabilir ve diğer 11 ve 12 numaralı bitler daima 0 dır.USB 2.0 cihazlarda ise 11 ve 12 numaralı bitler full-speed bir uçnoktanın mikroçerçevede desteklediği ek işlem sayısını tanımlar ve aşağıdaki değerlerden birini alır;

Değer	İlave
0x00	İlave Yok(Her mikroçerçeve de bir işlem olur)
0x01	İlave Var 1(Her mikroçerçeve de iki işlem olur)
0x02	İlave Var 2(Her mikroçerçeve de üç işlem olur)
0x03	Ayrılmış(Rezerve)

bInterval
Bu alanın cihaz'a göre birkaç anlamı vardır.Bir cihaz'ın uçnoktası kesme transferini destekliyorsa bu alan bu uçnoktanın yoklanması için maximum gecikmeyi, eğer cihaz izokron transfer'i destekliyorsa yoklanma aralığını ve cihaz yüksek hızlı OUT-Kontrol transferini kullanıyorsa bu alan uçnoktanın gönderilen paketlere kaç kere NAK ile yanıt verebileceğini belirler.
Düşük hızlı bir cihazda bInterval değeri milisaniye değerindedir ve 10 ile 255 arasında bir değer alabilir.Tam hızlı kesme ve 1.x izokron uçnoktalar için bu alan milisaniye cinsinden 1-255 arası değer alabilir.Spesifikasyon tam hızlı izokron uçnoktalar için bu değerin 1 olması gerektiğini söylemektedir.Izokron tam hızlı 2.0 cihazlar için bu değer 1 ile 18 arasında olabilir ve $2^{bInterval-1}$ şeklinde hesaplanır.Böylece 1ms ile 32768 sn arası değerler verilebilir.Yüksek hızlı uçnoktalarda ise bu değer 125 us'lik birimlerle ölçülür ve bir çerçeve genişliğindedir.1 ile 16 arası değerler verilerek 125 us ile 4096 sn arası bir aralık belirlenmiş olunur.
Yüksek hızlı OUT ve kontrol uçnoktaları için bu alan gönderilen isteklere maximum verebilecek NAK adedini belirler.Örneğin PC cihaza bir paket gönderdiğinde cihaz bu paketi herhangi bir sebepten dolayı almak istemiyorsa NAK ile yanıt verir.PC bunun üstüne tekrar istekte bulunacaktır.İşte bu durumda bu alan PC'ye yanıt olarak verilebilecek maximum NAK sayısını saklar ve 0 ile 255 arasında bir değer alabilir.
Tam hız yığın ve kontrol transferlerde bu alan dikkate alınmaz.

String Tanımlayıcısı / *String Descriptor*
Bu tanımlayıcı cihaz için tanımlanmış string bilgilerini içerir ve her madde için ayrı ayrı tanımlanır.Örneğin PC Aygıt Tanımlayıcısını okuduktan sonra iManufacturer ve iProduct alanlarında tanımlı index değerlerini kullanarak ilgili String Tanımlayıcısını ister.Tüm maddeler için tanımlanmış String Tanımlayıcıları bir dizi şeklinde saklanır.Bu dizideki String Tanımlayıcılarının yerleşimi diğer tanımlayıcılarda belirtilen index değerlerine göre olmalıdır.Örneğin Aygıt Tanımlayıcısında Üretici string index'i yani iManufacturer 1 ise Üretici ismi için tanımlanmış String Tanımlayıcısı dizinin 1. elemanında saklanmalıdır.PC istediği String Tanımlayıcısını Get_Descriptor ile ister.Bu isteği ileten paketteki wIndex alanı(daha sonra inceleyeceğiz) istenen String Tanımlayıcısının index'ini gösterir.Bizde bu index'i kullanarak oluşturduğumuz dizideki index'e karşılık gelen Tanımlayıcıyı döndürürüz.Tablo-7'de String Tanımlayıcısının alanları gösterilmiştir.

OffSet	Alan	Boyut	Açıklama
0	bLength	1	Bu açıklayıcının byte olarak uzunluğu
1	bDescriptorType	1	Açıklayıcı tipi.Bu alan sabit ENDPOINT(0x05)
2	bString, wLangId	Belli değil	Bu alan diğer açıklayıcılar için karakter katarı Ya da dil ID'sini saklar.

Tablo-7) String Tanımlayıcısı

bLength
Bu alan String Tanımlayıcısının toplam boyutunu tutar.Baştan iki alan birer byte, bString alanının boyutu ise belirlenen karakter katarına göre değişir.

bDescriptorType
Bu alan Tablo-2'de gösterilen STRING(0x03) değerini alır.PC bu tanımlayıcının String Tanımlayıcısı olduğunu bu değer ile algılar.

bString, wLangId
Bu alan cihaz ile ilgili karakter katarlarını tutar.Her String Tanımlayıcısının kendine ait bir index'i vardır ve diğer Tanımlayıcılar bir string tanımlayacaksa belirli bir alanında sadece index'ini tanımlar.Böylece PC ilgili tanımlayıcı ile ilgili bir karakter katarı okumak istediğinde alanlarından birinde belirttiği index değerine bakar ve bu index'e göre ilgili String Tanımlayıcısını seçer.Bunu PIC Frimware'ını incelerken daha iyi anlayacaksınız.
String Tanımlayıcılarını tutan dizinin ilk elemanı olan String Tanımlayıcısı 0 .eleman olduğundan bu alan Dil ID'sini göstermelidir.

Görüldüğü gibi tanımlayıcılar cihaz'ın geneli hakkında bilgiler tutan, belirli kurallar formatında dizilmiş byte dizeleridir.Bu byte'ların bu kurala göre dizilmesinin sebebi PC'nin gönderdiği istekler karşısında gönderilen byte dizelerinin bu formatta değerlendirilmesidir.Örneğin PC'ye istediği Aygıt Tanımlayıcısını 0x18, 0x05, 0x98..... şeklinde gönderdiğimizde bu formata göre 0. byte'ın bLength olduğu, 1. byte'ın bDescriptorType olduğunu bilir ve gerekli işlemleri yapar.

buffer[bLength] = 0x18, buffer[bDescriptorType] = 0x05 gibi..

Artık tanımlayıcıların ne olduğunu anladığınıza göre bu tanımlayıcıların hangi istekler karşısında hangi işlem basamaklarında ve hangi şekilde gönderildiğini incelemeye geçebiliriz.Daha önce listeleme aşamasının yürütülmesinde ve buna bağlı olarak ilgili tanımlayıcıların okunmasında kontrol transfer'lerine başvurulduğunu belirtmiştik.Bu yüzden bu transfer türünü, her işlem evresinde veri paketlerinin yapısını daha detaylı bir şekilde incelemeliyiz.

Kontrol Transferi'nin Detayları

Kontrol transfer'lerini Transfer türleri ve İçerikleri başlığı altında 3.bölümde incelemiştik.Fakat bu başlık altında sadece işlem basamakları ve her basamakta gönderilen paket yapıları incelenmişti.Şimdi bu bölümde bu paketlerin içerdiği bilgileri ve alanlarını inceleyeceğiz.

İşlem Evreleri

Daha önce kontrol transferlerinin üç basamaklı olduğunu söylemiştik.Bunlar SETUP, VERI ve STATUS basamaklarıydı.Her basamakta en az üç paket gönderiliyordu.Bunlar ise JETON, VERI ve ELSIKIŞMA paketleriydi.Şimdi bu basamakları ve paketleri daha detaylı inceleyelim.

Setup Basamağı

Bu basamak işlemin bir konfigrasyon olduğunu cihaza belirtmek için yürütülür.Her cihaz Setup basamağını tanımak ve ACK ile yanıtlamak zorundadır.Şimdi Setup basamağında iletilen JETON, VERI ve ELSIKIŞMA paketlerinin içeriklerini inceleyelim.

Jeton Paketi

Packet ID içeriği : SETUP(0x0D)
Paketin kaynağı : PC
Ek veriler : Cihaz ve uçnokta adresi
Paketin Anlamı : Bu paket cihaz'a işlem basamağının SETUP olduğunu anlaması için yollanır.

Veri Paketi

Packet ID içeriği : DATA0
Paketin kaynağı : PC
Ek veriler : bmRequestType, bmRequest, wValue, wIndex, wLength
Paketin Anlamı : Bu paket isteğin gerektirdiği bilgileri içerir.Cihaz bu bilgileri yorumlayarak en doğru verileri PC'ye göndermek veya uygun bir yanıtla çevirmek zorundadır.Bu yüzden bu veriler de belirli bir formatta dizilmiş bir byte dizesi olarak cihaz gönderilir.Şimdi bu alanları inceleyelim.

bmRequestType

Bu alanın uzunluğu bir byte'dır ve VERI BASAMAĞI'ndaki veriyi gönderecek kaynak bilgisini, isteğin tipini ve alıcı cihazı ifade eder.7.bit yön bitidir.Yani VERI BASAMAĞININ yönünü ifade eder.Bu bitin olması verinin cihaz'da PC'ye(IN), 0 olması ise PC'den cihaza(OUT) gideceğini bildirir.Bit 6 ve 5 isteğin tipini belirler ve aşağıdaki değerleri alırlar;

Değer	Istek tipi
0x00	Standart(USB'nin 11 standart isteği)
0x01	Sınıf(USB sınıfı için tanımlanmış özel istekler)
0x02	Üretici(Üretici tarafından tanımlanmış özel istekler)

4'ten 0'a kadar olan bitler ise isteğin cihazın hangi elemanına yönelik olduğunu gösterir ve aşağıdaki değerlerden birini alır;

Değer	Eleman
0x00	İstek doğrudan cihaza yönelik
0x01	İstek bir arabirime yönelik
0x02	İstek bir uçnoktaya yönelik
0x03	İstek cihaz içindeki bir elemana yönelik

bRequest
Bu alanın uzunluğu bir byte'dır ve isteği ifade eder.Bu alanın alacağı değer bmRequestType'ın 6 ve 5 bitleri ile belirlenir.Örneğin bu bitler 0x00 ise bu alan USB için tanımlanmış onbir standart istekten birini ifade edecektir.Bu alan 0x01 ise sınıfa özgü bir isteği, 0x02 olması durumda ise üretici tanımlı bir istek olduğunu belirtecektir.

wValue
Bu alan iki byte'dan oluşur ve PC isteğin gerektirdiği bilgiyi bu alanda gönderir.Örneğin PC bir Set_Address isteği gönderdiğinde bu alan içeriği ADRES bilgisi olacaktır.Bu gibi durumlarda VERI BASAMAĞINA gerek kalmaz.

wIndex
Bu alan iki byte'dan oluşur ve farklı amaçlarla kullanılır.Daha öncede bahsedildiği gibi bir String Tanımlayıcısı istenirken buradaki index değerine göre ilgili String Tanımlayıcısı gönderilir.Örneğin Manufacturer, yani üretici firma string'i, String Tanımlayıcı dizesinin 2. offset'inde olduğunu düşünelim.Bu durumda Aygıt Tanımlayıcısı'nın iManufacturer alanını 2 olarak belirlemeliyiz.Çünkü PC Aygıt Tanımlayıcısında bu değeri okuduktan sonra Ürtetici String'ini okumak için Get_Descriptor isteğine başvuracak, bu istekte 2 NUMARALI(wIndex = 2) STRING TANIMLAYICISI'nı isteyecektir.
Bu alan isteğin gerektirdiği bilgiye göre değişiklik gösterir.Örneğin STRING alma işlemlerinde işleyiş yukarıda anlatıldığı gibi yürürken, bir arabirim özelliği set edilirken(Set_Interface) bu alanın içeriği arabirim numarasını ifade eder ve 0 ile 7 arası bitler kullanılır.Eğer bir uçnokta özelliği set ediliyorsa(Set_Feature) bu durumda wIndex alanı içeriği uçnokta numarasını ifade eder ve 0 ile 3 arası bitler kullanılır.

wLength
Bu alan iki byte'dan oluşur ve veri basamağında gönderilecek veya alınacak veri adedini ifade eder.Bu değer 0 ise Veri basamağında işlem olmaz.

Elsıkışma Paketi

Paket ID içeriği	:ACK
Paketin Kaynağı	:USB Cihaz
Ek veriler	:Bu pakette ek veriler bulunmaz.Sadece PID bulunur.
Paketin Anlamı	:Setup Paketini ve Veri paketini sorunsuz alan bir cihaz elsıkışma paketinde ACK döndürerek, işlemleri onaylar ve SETUP BASAMAĞI tamamlanmış olunur.USB cihaz bir hata tespit etmesi halinde elsıkışma yollamayabilir.

Veri Basamağı
Bu basamakta Setup basamağındaki isteğin gerektirdiği bilgi taşınır veya PC'den cihaza bilgi transfer'i yapılır.Yani bu basamak mutlaka IN veya OUT işleminden oluşur.Bu basamaktaki işlemin, yani verilerin cihaz'a doğru'mu olacağı veya cihaz'dan PC'ye doğru mu olacağı SETUP basamağındaki Veri paketinde bulunan bmRequestType'ın 7 biti ile cihaza önceden bildirilir.İsteğin içeriğini ve bu biti değerlendiren cihaz ya veri basamağında veri alır, ya da PC'ye gerekli verileri iletir.
Örneğin cihaz Get_Descriptor isteği aldığı zaman veri basamağında bu isteğin gerektirdiği bilgiler taşınır.(IN)Eğer PC bir HID sınıfı isteği olan Set_Report'u gönderirse bu sefer cihaz bu basamakta PC'den veri alır(OUT)
Her işlem'de taşınacak verilerin büyüklüğü uçnokta tanımlayıcılarında tanımlı, uçnoktanın maximum paket büyüklüğü ile sınırlıdır.Aynı zamanda gönderilecek veriler tek bir paket'e sığmıyorsa birden fazla işlem evresi gerçekleşir.Her işlem evresinden biraz öncede bahsedildiği gibi uçnokta'nın maximum paket büyüklüğü kadar veri gönderilir.Bu işlem sayısı ise gönderilecek verilerin adedi ile uçnokta'nın maximum paket büyüklüğünün bölünmesi ile bulunur.Örneğin cihaz'a 18 byte veri gönderilecek ve uçnokta'nın maximum paket büyüklüğü 8 ise bu durumda bu basamakta gerçekleşecek işlem sayısı 18/8'den 3 olacaktır.Dikkat ederseniz sonuç en yakın sayıya yuvarlanmıştır.
Eğer setup basamağındaki veri paketinde tanımlı wLength alanı 0 ise Veri basamağı gerçekleşmez.Bu gibi durumlar olabilir çünkü PC bazen cihaza veri göndermek için bu basamağı değil Setup basamağının veri paketinde tanımlı wValue alanını kullanır.Örneğin PC cihaz'a bir adres ataması yaparken Set_Address isteğine başvurur ve cihaz için tayin ettiği adres bilgisini Setup basamağının veri paketinde tanımlı olan wValue alanında gönderir.Bu durumda wLength alanı 0 olur çünkü veri basamağına gerek yoktur.
Şimdi bu basamaktaki Jeton, Veri ve Elsıkışma paketlerinin içeriğini inceleyelim.

Jeton Paketi

Paket ID içeriği :Eğer isteğin içeriği PC'ye veri gönderileceğerini belirtiyorsa IN aksi halde OUT
Paketin Kaynağı :PC
Ek veriler :Cihaz ve uçnokta'sının adres bilgileri
Paketin Anlamı :Bu paket transfer işleminin hangi yöne olduğunu belirtmek için gönderilir.Böylece alıcı taraf işlemin bir IN veya OUT olduğunu tespit etmiş olur.

Veri Paketi

Paket ID içeriği :Her zaman gönderilen ilk paketin PID alanı DATA1'dir.Bundan sonraki işlemlerde ise DATA0-DATA1.... şeklinde devam eder.
Paketin Kaynağı :Eğer Jeton paketinin PID alanı OUT ise PC, IN ise cihaz
Ek veriler :Gönderilen veri
Paketin Anlamı :Bu pakette SETUP basamağında belirtilen isteğin gerektirdiği bilgiler tranfer edilir veya PC cihaz'a veri gönderir.Gönderilen verinin büyüklüğü yine SETUP basamağındaki wLength alanı ile belirlenir.

Elsıkışma Paketi

Paket ID içeriği :Cihaz isteklere karşı ACK, NAK, STALL ve NYET(high-speed) gönderebilirken, PC sadece ACK döndürür.
Paketin Kaynağı :Bu paketi verileri alan taraf gönderir.Eğer Jeton paketinin PID alanı IN ise bu paketi PC, eğer OUT ise cihaz gönderir.
Ek veriler :Bu pakette ek veriler bulunmaz.Sadece PID bulunur.
Paketin Anlamı :Bu paket alınan verilere karşı alıcı taraf tarafından emir tekrarı(ACK) yapmak için gönderilir.Fakat alıcı taraf bir hata tespit ederse elsıkışma paketi göndermeyebilir.

Status Basamağı

Bu basamak Elsıkışma paketine benzer fakat, sadece bir paketin başarılı bir şekilde alınıp alınmadığını değil tüm transferin başarılı olup olmadığını belirtir.Her cihaz hangi basamağı işliyor olursa olsun PC'nin isteği üzerine diğer basamakları terkedip STATUS basamağını yanıtlamak zorundadır.Şimdi bu basamakta gönderilen Jeton, Veri ve Elsıkışma paketlerini inceleyelim.

Jeton Paketi

Paket ID içeriği :Veri basamağının alıcısı cihaz ise (OUT) bu alan IN aksi halde OUT olur.
Paketin Kaynağı :PC
Ek veriler :Cihaz ve uçnokta'sının adres bilgileri
Paketin anlamı :Bu paket bu basamağın yani Status basamağının veri paketi yönünü belirlemek için için gönderilir.

Veri Paketi

Paket ID içeriği :DATA1
Paketin Kaynağı :Bu basamağın yani Status Basamağının Jeton paketi PID'ı IN ise cihaz, OUT ise PC veri gönderir.
Ek veriler :Transferin başarısını 0 uzunluklu bir veri paketi belirler.0 uzunluklu bir veri paketinde sadece PID ve CRC bitleri bulunur, veri bitleri bulunmaz.Status basamağında veriyi PC gönderiyorsa sadece 0 uzunluklu bir paket gönderir.Cihaz gönderiyorsa 0 uzunluklu bir paket, NAK ya da STALL gönderir.
Paketin Anlamı :Bu paket biraz öncede açıklandığı eğer bir hata yoksa sıfır uzunlukta bir paket içerir. Bu tüm transferin başarılı olduğunu gösterir.Sıfır uzunluklu bir paket başka amaçlarda da kullanılır.Örneğin istekler iletilirken önden gönderilen bir sıfır uzunlukta paket, isteğin iletiyor durumda olduğunu ifade eder.

Elsıkışma Paketi

Paket ID içeriği :Veri paketini cihaz gönderdiği ise PC emir takrarı yapar ve PID alanı ACK olur. Eğer verileri PC gönderdi ise yani alıcı taraf cihaz ise bu alan ACK, NAK ve STALL olabilir.
Paketin Kaynağı :Eğer bu basamağın yani Status basamağının veri paketi alıcısı PC ise Elsıkışma paketini de PC gönderir aksi halde cihaz gönderir.

Ek veriler :Bu pakette ek veri bulunmaz.Sadece PID'dan ibarettir.
Paketin Anlamı :Bu paket Status basamağının veri paketini alan alıcı tarafından gönderilir ve tüm transferin son paketidir.

Görüldüğü gibi her basamakta iletilen paketlerin de daha önce anlatılan tanımlayıcılar gibi belirli bir düzeni ve yerleşimi vardır.Aslında tüm transfer'ler bir byte dizisi halinde yolculuk yapar.Fakat PC'nin ve cihazın hangi byte'ın ne anlama geldiğini bilmesi ve buna göre işlem yapması gerekir.Bu yüzden tüm veriler belirli bir kalıp şeklinde tasarlanmıştır.Bu şekilde bu kurala göre tasarlanmış bir cihaz kodu, hangi sıradaki byte'ın hangi anlama geldiğini, hangi sıradaki byte'ın hangi bitinin ne anlama geldiğini bilmiş olur.Örneğin Setup basamağının veri paketinde gönderilen byte'ların yerleşimini incelemiştik.Cihaz bu yerleşimi bildiğinden ilk byte bmRequestType ikinci byte'ın bmRequest ... olduğunu bilir.Böylece birinci byte'ın en değerlikli biti (7.bit) transfer yönünü gösterir diyerek gerekli işlemleri ve hazırlıkları yapabilir.Şimdi bu kalıpların istekler gönderilirken hangi değerler ile doldurulduğuna bakalım.
Gönderilen her isteğe göre bu yapıda olan byte'lar değişiklik gösterecektir.Böylece cihaz farklı istekleri ve işlemleri yine bu yapıyı baz alarak değerlendirebilecektir.

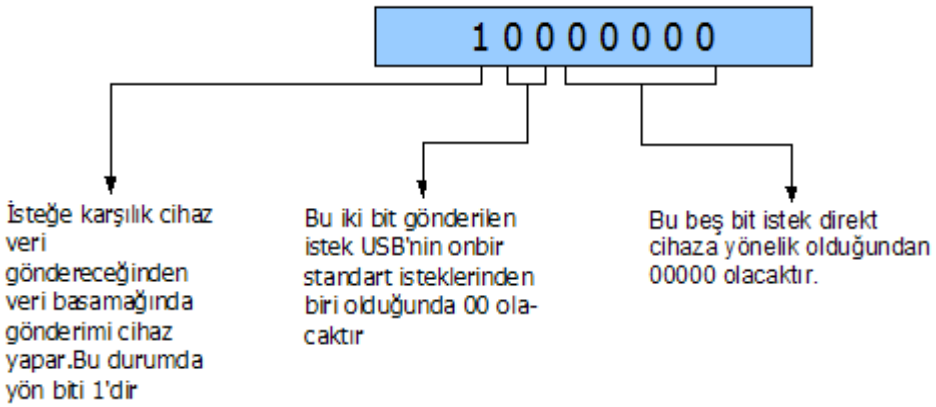
Standart İstekler

Daha öncede bahsedildiği gibi bir USB cihaz PC'deki portlardan herhangi birine bağlandığı zaman PC cihazı tanımak, sınıflandırmak ve yeteneklerini öğrenmek için bir dizi istekte bulunur.Cihaz bu isteklerin bazılarını desteklemeyebileceği gibi, bazılarınıda mutlaka desteklemek zorundadır.Cihaz desteklemediği istekler olsa bile mutlaka STALL ile yanıt vermelidir.
Cihazların listelenmesi için tanımlanmış toplam onbir istek bulunur ve bunlar 0x00 ile 0x0C arasındadır.Bu istekler cihaza gönderilirken SETUP basamağında yukarıda bahsedilen yapıya uygun olarak gönderilir.Şimdi bu istekleri daha yakından inceleyelim.

Get_Descriptor (0x06)

Açıklama:
Bu istek cihazdan herhangi bir tanımlayıcıyı göndermesini bildirmek amacıyla SETUP basamağında gönderilir.

bmRequestType İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan, bu isteğin gönderilmesi halinde aşağıdaki bit yapısında olur;



Şekil-33) Get_Descriptor için bmRequestType içeriği

bRequest İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bmRequestType'ın 6.ve 5.bitleri 00 olduğundan USB'nin onbir standart isteklerinden birini içerir.Get_Descriptor için bu istek numarası 0x06'dır.

wValue İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın yüksek byte'ında tanımlayıcı tipi düşük byte'ında ise tanımlayıcı değeri bulunur.

Örneğin bu istek ile String Tanımlayıcısı isteniyorsa yüksek byte 0x03(Tablo-2 Tanımlayıcı tablosuna bakın)değerini alacaktır.

wIndex İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan String Tanımlayıcısı hariç daima 0'dır.String Tanımlayıcısında ise daha önce anlatıldığı gibi istenen String'in index'ini içerir.

wLength İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan gönderilmesi istenen, isteğin gerektirdiği verilerin toplam uzunluğunu gösterir.Bu değer eğer cihazın uçnoktasının gönderebileceği veri boyundan büyükse cihaz uçnokta büyüklüğü kadar veri gönderir, ardından sıfır uzunluklu bir paket daha gönderir.Bu durumda PC kalan verileride isteyecektir.

Bu istek ile alınmak istenen tanımlayıcılar Tablo-2'de listelenmiştir.Konfigrasyon tanımlayıcısının istenmesi ile bu tanımlayıcıyı takiben arabirim ve uçnokta tanımlayıcılarıda gönderilmelidir.(Şekil-32'deki hiyerarşik yapıya bakın)

Set_Descriptor (0x0B)

Açıklama:
Bu istek cihaz'a belirli bir künye ekler veya herhangi mevcut bir künyeyi günceller.Direkt olarak cihaz yazılımını değiştirdiğinden birçok cihaz tarafından desteklenmez.

bmRequestType İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bu istekde de Şekil-33'daki yapıda olacaktır.

bRequest İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bmRequestType'ın 6.ve 5.bitleri 00 olduğundan USB'nin onbir standart isteklerinden birini içerir.Set_Descriptor için bu istek numarası 0x0B'dir.

wValue İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın yüksek byte'ında tanımlayıcı tipi düşük byte'ında ise tanımlayıcı değeri bulunur.

wIndex İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan String Tanımlayıcısı hariç daima 0'dır.String Tanımlayıcısında ise daha önce anlatıldığı gibi istenen String'in index'ini içerir.

wLength İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan PC'nin güncellemek ya da eklemek istediği tanımlayıcının byte'larının uzunluğunu içerir.

Get_Configuration (0x08)

Açıklama:
Bu istek cihazdan daha önce set edilen konfigrasyon değerini almak için kullanılır.Bu değer aslında Konfigrasyon Tanımlayıcısının bConfigurationValue alanında saklanan değerdir.PC cihazı listelerken, eğer cihaz belirtilen şartları sağlıyorsa bu değeri set edecektir.(Set_Configuration'a bakın)

bmRequestType İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bu istekde de Şekil-33'daki yapıda olacaktır.

bRequest İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bmRequestType'ın 6.ve 5.bitleri 00 olduğundan USB'nin onbir standart isteklerinden birini içerir.Get_Configuration için bu istek numarası 0x08'dir.

wValue İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın değeri bu istek için sıfır'dır.

wIndex İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın değeri bu istek için sıfır'dır.

wLength İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan istenen konfigürasyon değerinin uzunluğunu içerir. Konfigrasyon Tanımlayıcısının bConfigurationValue alanı 1 byte olduğundan bu uzunluk değeride bu alanda 1 byte olacaktır.

Set_ Configuration(0x09)

Açıklama:
Bu istekle PC, cihaz tarafından tanımlanan ve okunan bilgilerin uygun olduğunu kabul eder ve Konfigurasyon Tanımlayıcısında belirtilen Konfigurasyon değerini set eder.

bmRequestType İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bu istekde de Şekil-33'daki yapıda olacaktır.

bRequest İçeriği:
SETUP basamağının Veri paketinde bulunan bu alan bmRequestType'ın 6.ve 5.bitleri 00 olduğundan USB'nin onbir standart isteklerinden birini içerir.Set_Configuration için bu istek numarası 0x09'dur.

wValue İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın düşük byte'ında cihaz tarafından belirlenmiş Konfigrasyon değeri bulunur.Cihaz bu istekdeki bu alanı okur ve önceden belirlediği bir alanda bu değeri saklar.Sakladığı bu değer daha sonra Get_Configuration ile okunabilir.
Eğer cihaza daha önce bu istek gönderilmemişse cihaz içindeki bu alan değeride 0 olacaktır.Bu durumda cihaz ADDRESS durumundadır.

wIndex İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın değeri bu istek için sıfır'dır.

wLength İçeriği:
SETUP basamağının Veri paketinde bulunan bu alanın değeri bu istek için 0'dır.

Bu bölüme kadar yazılanlar USB konusuna ve işleyişine giriş niteliğindeydi.Bundan sonraki bölümlerde isteklerin diğer çeşitlerine ve İÇİNDEKİLER'de belirtilmiş diğer konulara yer verilecektir.