

YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Türkçe Fonetiklerle Ses Sentezi

YÜKSEK LİSANS TEZİ

H. Cahit Terzioğlu

1989

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

	R 368
Kot	52
Alındığı Yer	Fen-Bil. Ens.
Tarih	13.4.95
Fatura	
Fiyatı	15.000.-
Ayniyat No	1-4
Kayıt No	51030
UDC	
Ek	

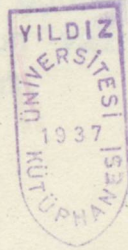
YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TÜRKÇE FONETİKLERLE SES SENTEZİ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜH. H.CAHİT TERZİOĞLU

İSTANBUL, 1989



ÖNSÖZ

Bilgisayarlarla yapılan işler gün geçtikçe hem çeşit olarak hem de boyut olarak artmaktadır. Buna insanların kendilerine benzer bir makinanın gerçekleştirme rüyası da eklenince, bilgisayarlarla yapılan veya bilgisayara yaptırılan işler başka bir anlam kazanmaktadır. Tabii bilgisayarın bir elektronik yığın olduğu unutulmadan insan aklının geliştirdiği bazı yardımcı unsurları kullanarak insanların çeşitli fonksiyonları taklit edilebilmektedir.

Bu yüzden araştırmacılar insanların konuşma, işitme, kavrama gibi olayları bilgisayara veya bilgisayar yardımı ile robotlara yaptırmak için çeşitli çareler aramaktadırlar. Unutulmamalıdır ki, yapılan bu çalışmalar gerçeğe ancak bazı toleranslar dahilinde yaklaşabilmektedir.

Bu projede amaç, Türkçe yazılmış bulunan bir metni Türkçe fonetikler yadımı ile bilgisayar tarafından söylenmesidir.

Bu projenin hazırlanmasında değerli yardımlarından dolayı Bilgisayar Bilimleri Bölüm Başkanı Sayın Prof. M.Yahya KARSLIGİL'e ve aynı bölümün öğretim üyelerinden Sayın Doç. Kırkor HARUTUNYAN'a, bu raporun yazılmasında emeği geçen tüm arkadaşlarıma teşekkür ederim.

YILDIZ ÜNİVERSİTESİ
İSTANBUL, 1989

H.Cahit TERZİOĞLU

İÇİNDEKİLER

Özet

Abstract

1. Ses

1.1. Ses nedir?	1.1
1.1.1. Tanımı ve içeriği	1.1
1.1.1.1. Dillerdeki sesler	1.1
1.1.2. Seslerin çıkarılması, ses yolu, ses cihazı	1.2
1.1.2.1. Ses geçidi	1.2
1.1.2.2. Ses telleri	1.4
1.1.2.3. Küçük dil, ağız yolu, burun yolu	1.4
1.1.3. Sedalı sesler, sedasız sesler	1.5
1.1.4. Vokal	1.6
1.1.4.1. Vokallerin doğası ve yeteneği	1.7
1.1.4.2. Vokallerin uzunluğu ve kısalığı	1.7
1.1.5. Konsonant	1.7
1.1.5.1. Konsonantların doğası ve özellikleri	1.8
1.2. Türkçedeki sesler	1.8
1.2.1. Harf, alfabe, sesler	1.8
1.2.2. Vokaller	1.8
1.2.2.1. Vokallerin sınıflandırılması	1.9
1.2.2.1.1. Kalın vokal, ince vokal	1.9
1.2.2.1.2. Geniş vokal, dar vokal	1.10
1.2.2.1.3. Düz vokal, yuvarlak vokal	1.10
1.2.2.1.4. Uzun vokal, kısa vokal	1.11
1.2.2.1.5. Sonuç	1.11
1.2.3. Konsonantlar	1.12
1.2.3.1. Konsonantların sınıflandırılması	1.13
1.2.3.1.1. Sedalı konsonant, sedasız konsonant	1.13
1.2.3.1.2. Oluşum noktalarına göre konsonantlar	1.13
1.2.3.1.3. Sürekli konsonantlar, süreksiz	

konsonantlar	1.14
1.2.3.1.4. Geniz konsonantı, ağız	
konsonantı	1.15

2. Macintosh ve MacinTalk

2.1. Macintosh	2.1
2.1.1. Macintosh Plus'un teknik özellikleri	2.1
2.1.2. Macintosh İşletim Sistemi	2.2
2.2. MacinTalk	2.2
2.2.1. Genel bilgi	2.2
2.2.2. MacinTalk sürücüsünü kullanma	2.2

3. Program hakkında bilgi

3.1. Genel bilgi	3.1
3.2. Türkçe metnin hazırlanması	3.1
3.3. 'Konuş' adlı programın çalıştırılması	3.2

Sonuç

Programlar

Konuş.Pas	Pr1
Memtypes.ipas	Pr2
OsIntf.ipas	Pr3
Quickdraw.ipas	Pr4
Speech.ipas	Pr5
ToolIntf.ipas	Pr6

Kaynaklar

Özgeçmiş

ÖZET

ABSTRACT

Konusu Macintosh bilgisayarı ile Türkçe ses sentezleme olan bu projede, ana hedef olarak, Türkçe yazılmış bir metni bilgisayarın hoparlöründen ses olarak algılanmasıdır.

Sistemde TML Pascal ile derlenen Pascal program ve bu programın kullandığı MacInTalk Sound Driver (MacInTalk Ses Sürücüsü) vardır. MacInTalk Apple Corp.'nın Macintosh bilgisayarı ile verdiği yardımcı programdır. Bundan dolayıda kendisinde bulunan bazı sesler İngilizceye yatkındır.

İlk önce konuşturulması istenen metin bir Editor (Edit, MacWrite gibi) yardımı ile yazılır. Ancak bu metin 5120 byte (sekizlik) 'dan uzun olmamalıdır. Sonuna iki adet "##" konarak metin sonu belirlenir. Daha sonra "**Konuş**" isimli program çalıştırılır.

Metnin konuşma hali hoparlörden veya Macintosh'un arkasında bulunan hoparlör çıkışıdan yararlanılarak duyulur.

ABSTRACT

In this project which subject of it is Turkish speech synthesesis with Macintosh computer; main target is to convert the text which is written in Turkish, yo separate Turkish utterance.

In this system, there are the program which is compiled by TML Pascal and **MacinTalk Sound Driver** which is used by the program. MacinTalk which is given by Apple Corp; is one of the untility programs. For this reason, some utterances which is on that; are familiarity to English.

First of all, the text which is wanted to be spoken; is written by Editor (Edit, MacWrite etc.). But, the text is **no longer** than **5120 byte**. '##' is put end of text for limitation of text. After that, the program which is called '**Konuş**'; is been run.

Utterance of the text is heard from loudspeaker.

BÖLÜM 1

1. Sesler

1.1. Ses nedir?

1.1.1. Tanımı ve İçeriği

En küçük gramer birimlerinden en büyük birlik olan cümleye kadar dil meydana gelen bütün şekillerin bünyesinde ses vardır ve dediğimiz küçük ve en basit dil unsurları bunlardır. Bu unsurlar bazen tek başlarına çok defa da yan yana gelerek canlı canlı varlıkları, kavramları ve durumları karşılayan kelime dediğimiz dil birliklerini, dil unsurlarını, dil şekillerini meydana getirirler. Kelime Anlamı veya görevi ses ve sesler topluluğudur. Tek sestən oluşmadığı zaman **BÖLÜM 1** küçük gramer birlikleri, xdk ve ek sesler olarak karşımıza çıkar. Demek ki en küçüğünden en büyüğüne kadar bütün dil birlikleri seslerden oluşur. Yani, sesler dilin malzemesidir. Gramer birliklerinden en küçüğü tek sesli olur. Sesten daha küçük bir gramer birliği yoktur. Ses parçalanılamayan en küçük gramer birliğidir. O halde ses dilin en küçük parçasıdır. a, d, f, y, gibi.

1.1.1.1. Dilimizdeki sesler

İnsan ağızından çıkarabileceği sesler kısıtlı olduğu için, bütün dillerdeki sesler dillerin çeşitliliğine uygun düşecek kadar çok değildir. Seslerin büyük bir kısmı genel olarak dillerde ortaktır. Dilleri birbirinden ayıran şey temelde sesler olmayıp, seslerin meydana geldiği şekillerdir. Gramer birlikleridir. Aynı seslerden her dilde başka şekillerle birlikler yapılır. Seslerin büyük bir çoğunluğu her dilde aynıdır. Bazen her dilin kendisine özgü sesleri de yok değildir. Bir dilde kendisinde bulunmayan bir ses taşıyan yabancı kelime girerse, o kelimedeki yabancı ses dilin ona yakın bir sesle değiştirilir.



1.1.2. Seslerin çıkarılması, ses yolu, ses sırası

1. Sesler

1.1. Ses nedir?

1.1.1. Tanımı ve içeriği

En küçük gramer birliklerinden en büyük birlik olan cümleye kadar dili meydana getiren bütün şekillerin bünyesinde ses adını verdiğimiz küçük ve en basit dil unsurları bulunur. Bu unsurlar bazan tek başlarına, çok defa da yan yana gelerek canlı cansız varlıkları, kavramları ve durumları karşılayan kelime dediğimiz dil birliklerini, dil unsurlarını, dil şekillerini meydana getirirler. Kelime anlamlı veya görevli ses veya sesler topluluğudur. Tek sestən oluşmadığı zaman, kelimenin yapısında daha küçük gramer birlikleri, kök ve ek dediğimiz ses veya ses toplulukları bulunur. Demek ki en küçüğünden en büyüğüne kadar bütün dil birlikleri seslerden oluşur. Yani, sesler dilin malzemesidir. Gramer birliklerinin en küçüğü tek sesli olur. Sesten daha küçük bir gramer birliği yoktur. Ses parçalanamayan en küçük gramer birliğidir. O halde ses dilin en küçük parçasıdır. *a, d, l, y* gibi.

1.1.1.1. Dillerdeki sesler

İnsan oğlunun çıkarabileceği sesler kısıtlı olduğu için, bütün dillerdeki sesler dillerin çeşitliliğine uygun düşecek kadar çok değildir. Seslerin büyük bir kısmı genel olarak dillerde ortaktır. Dilleri birbirinden ayıran şey temelde sesler olmayıp, seslerin meydana getirdiği şekillerdir, gramer birlikleridir. Aynı seslerden her dilde başka şekiller, başka birlikler yapılır. Seslerin büyük bir çoğunluğu her dilde aynı olmakla beraber her dilin kendisine özgü sesleri de yok değildir. Bu yüzden, bir dile kendisinde bulunmayan bir ses taşıyan yabancı bir kelime girerse, o kelimedeki yabancı ses dilin ona yakın bir sesine çevrilir.

1.1.2. Seslerin çıkarılması, ses yolu, ses cihazı

1.1.2.1. Ses geçidi

Sesler akciğerlerden başlayıp ağız ve burunda sona eren solunum yolları ile bu yollar üzerinde sıralanmış organlar tarafından meydana getirilir. Seslerin söylenmesinde akciğerden itibaren sırayla nefes borusu, gırtlak, ağız boşluğu ve burun yolu bir ses geçidi olarak görev yaparlar. Bu ses geçidinin çeşitli noktalarında hareket edebilen, birbirine yaklaşıp uzaklaşabilen organlar vardır. Bu organlarında yardımı ile ses geçidi açılıp kapanmak, daralıp genişlemek yoluyla değişik şekillere girebilir. İşte ciğerlerden yükselen hava akımı dışarı çıkmak üzere ses geçidinden geçerken geçidin içindeki organları da harekete getirerek açılıp kapanması ve daralıp genişlemesi sonucunda sesler oluşur. Sıkışıp açılan geçitte hava akımı kendisine yol bulmak için karşısına dikilen organları zorlar, onlara çarpar, sürtünür, böylece seda haline geldikten sonra son olarak bir noktada karşısına çıkan bir kalıptan, bir engelden geçer ve ses haline gelerek dışarı çıkar. Dışarı çıkmak isteyen havanın ses geçidinde çarptığı ve sürtüldüğü yol ve organlar hançere, boğaz, ağız ve burun boşlukları ile ses telleri, küçük dil, damak, dil, dişler ve dudaklardır. *Şekil 1.3* te ses geçidinin şekli bulunmaktadır.

1.1.2.2. Ses telleri

Demek ki seslerin oluşması için her şeyden önce ciğerlerden bir hava akımının yükselmesi lazımdır. Ciğerlerin sıkışması ile yükselen bu hava akımı nefes borusundan geçerek o borunun sonunda, seslerin oluşması bakımından ilk önemli noktaya, seslerin oluşmasında çok büyük rolü olan bir noktaya gelir. Bu noktada ses telleri bulunur. Ses telleri adını verdiğimiz organ karşıdan karşıya atılmış teller halinde olmayıp, hançerenin sonunda karşı karşıya gelen yarım daire şeklinde iki dudakçıktan meydana gelmiştir. Bu dudakçıklar ön tarafta daima bitişik bulunurlar. Arka tarafta ise normal nefes ve bir çok seslerin oluşması sırasında aralıklı bulunurlar. Yani bir kısım seslerin oluşmasına ses telleri katılmaz, hava akımı arkadaki aralıktan geçerek yoluna devam eder. Diğer kısım seslerin oluşmasına ise ses telleri büyük ölçüde katılırlar. Ses tellerinin katılması gereken sesler çıkarılacaksa derhal dudakçıkların arka tarafı da kapatılır. Ciğerlerden gelen hava birleşmiş olan dudakçıklar arasından kendisine bir yol açmak üzere onları iter. Bu itme neticesinde hava akımının karşısına çıkmış olan ses telleri titrer, hava akımı onları titreterek aralarından geçer. Bu titreşimden bir kısmı seslerin oluşması için gerekli olan seda meydana gelir. Ses telleri bazan da, kapandıktan sonra titreşmez ve hava akımına hiç yol vermezler. Hava onlara çarparak sesi meydana getirir.

1.1.2.3. Küçük dil, ağız yolu, burun yolu

Seslerin oluşumunda ciğerlerden yükselen hava akımının ses geçidinde ses tellerinden sonra uğradığı ikinci önemli nokta ağız ve burun yollarının birleştiği noktadır. Bu noktada küçük dil bulunur. Küçük dil damağın arka tarafına, yumuşak damağın sonuna asılı bulunan ve hava akımı karşısında çok kolay hareket edebilen bir organdır. Küçük dil, gelen hava karşısında, oluşacak sese göre durum alır. Gelen havaya ağız yolunu açık tutması gerekince, yükselerek geniz yolunu kapatır. Havanın geniz yolundan çıkması gerektiren seslerde ise, aşağıya sarkarak arka tarafı yükselen dil

ile birlikte ağız yolunu kapatır ve havayı geniz yoluna gönderir. Geniz seslerinin sayısı çok azdır. Seslerin büyük bir kısmı ağız yolu açık olduğu zaman oluşurlar. Ağız yolunda seslerin oluşması için başlıca rolü oynayan organ dildir. Dil, gelen hava ile seda akımı karşısında, oluşacak sese göre çeşitli hareketler yapar, türlü şekillere girer. Kabarır, yayılır, damak ve dişlere dokunur. Böylece gelen hava veya seda akımını çeşitli noktalarda çeşitli kalıplardan geçirerek şekilli seda, yani ses haline getirir. Böylece hava veya seda akımı ses tellerinden dişlerin dışına çıkıncaya kadar bir çok sesler oluşmuştur. Bazı seslerde ise hava veya seda akımı dişleri geçtikten sonra bile henüz ses haline gelmiş olmaz. Bu seslerin oluşumunda dudaklar rol oynar. Gelen ve o zamana kadar karşısına bir engel çıkmamış olan akımı dudaklar kalıptan geçirerek ses haline getirirler.

1.1.3. Sedalı sesler, sedasız sesler

Oluşumlarında ses telleri titreşmeyen sesleri duyuran seda ise hava akımının hançereden dudaklara kadar ses geçidinde karşısına çıkan engellere çarpması veya sürtünmesinden ileri gelir. Böyle olduğu için, sedasını ses tellerinin titreşiminden alan seslerin oluşması sırasında hem hava akımı hafif, hem de ses geçidinin havanın karşısına çıkardığı engeller zayıf olur; buna karşılık, sedasını çarpma ve sürtünmeden alan seslerin oluşmasında hem hava akımı, hem de ses geçidinin bu akım karşısına çıkardığı engeller kuvvetli olur. Engellerin zayıf veya kuvvetli olması demek ses yolunun az veya çok kapalı olması, hava akımı için rahat veya rahat olmayan bir geçidin bulunması, ses geçidindeki organların normal durumlarından az veya çok farklı hale geçmeleri, gergin veya gevşek olmaları gerekmektedir. Şüphesiz sedalarını ses tellerinden alan sesler de, çarpma veya sürtünmeden alan sesler de kendi aralarında bu bakımdan birbirinden farklıdır. Bu farklar çarpma ve sürtünme ile oluşan seslerde az, oluşumlarında ses telleri titreşen seslerde ise çok fazladır. Öyle ki titreşimli seslerin bir kısmı diğer kısmından çok büyük ölçüde ayrılarak, organların gergin veya gevşek olması bakımından aralarında çok

fark olduđu halde, çarpma ve sürtünmeyle meydana gelen sesler sınıfına girer.

Demek ki seslerin bir kısmı oluşumları sırasında belirli hiç bir engele takılmadan, hiç bir zorluğa uğramadan çıkan sesler; diğerleri ise oluşumları sırasında zayıf veya kuvvetli fakat belirli bir engele takılarak, belirli bir takıntıya uğrayarak çıkan seslerdir. Takıntıya uğrayarak çıkan seslerin bir kısmı ile takıntıya uğramadan çıkan sesler sedalarını ses tellerinden, takıntıya uğrayarak çıkan seslerin geri kalanı da sedalarını sürtünme ve çarpmadan alırlar.

Sesler için gerçek seda ses tellerinden çıkan sedadır. Ancak bu çeşit sedanın değişik ses perdeleri bulunur, konuşmada veya melodide ses perdesi ancak bu seda ile yükselip alçalabilir. Çünkü ses perdesinin değişmesi ses tellerindeki titreşimin frekansına, ses tellerinin saniyedeki titreşim adedine bağlıdır. Ses tellerinin titreşiminden değil de çarpma ve sürtünmeden doğan seslerin perdesi değiştirilemez. O halde, gerçek sedalı sesler sedalarını ses tellerinden alan seslerdir. Bu sebeple oluşumlarında ses telleri titreyen seslere *sedalı*, oluşumlarında ses telleri titremeyen seslere *sedasız* diyoruz. Demek ki takıntıya uğramadan çıkan bütün seslere takıntıya uğrayarak çıkan seslerin bir kısmı sedalı, takıntıya uğrayarak çıkan seslerin diğer kısmıda sedasızdır.

1.1.4. Vokal

Oluşumları sırasında ses geçidinde belirli hiç bir takıntıya uğramayan, hiç bir engelle karşılaşmayan seslere *vokal* adı verilir. Vokaller oluşurken ses yolu tamamiyle açık bulunur. Gelen hava ve seda akımı rahat bir yolla dışarı çıkar. Yalnız, yolun açıklık derecesi vokallerin çeşitlerine göre değişir. Bu değişiklik çenelerin az veya çok açılması, dilin kabarıp alçalması ve dudakların yuvarlaşıp düzleşmesi ile temin edilir. Vokallerin oluşumunda oluşum noktalarına göre dilin arkası, ortası veya ucu kabarır. Fakat bu kabarmalar hiç bir zaman geçidi kapatacak

şekilde olmaz ve ortada rahat bir yol kalır. Vokallerin dışındaki seslerde boğumlanmayı temin etmek üzere oluşum noktalarında organlar, dil, damak, dişler, dudaklar v. b. zayıf veya kuvvetli olarak birbirleriyle temasa gelirler. Esasen oluşum noktaları da bu temas ve yaklaşma noktalarının zirveleridir. Vokallerin oluşumunda açık olan dudaklar ise düz veya yuvarlak şekiller alır. Bu düzlük ve yuvarlaklık da vokallere göre dar veya geniş olur. Sonra, vokaller sedalı seslerdir. Yani, oluşumlarının başlangıcında ses telleri de harekete geçer.

1.1.4.1. Vokallerin doğası ve yeteneği

Vokallerin bu yumuşak doğaları, oluşumlarında organların gösterdiği bu etkisiz hal onları diğer seslerden daha üstün bir duruma sokar, onlara diğer seslerin yanında daha önemli görevler yükler. Vokaller tek başlarına hece, kelime, kök ve ek olabilirler. Böylece bir vokal bazan bir taraftan bir ses, diğer taraftan anlamı veya görevi olan bir gramer birliği olarak çifte kimliğe sahip olur. Vokal dışında kalan seslerde ise bu yetenek yoktur. Onlar tek başlarına sadece ek olabilirler.

1.1.4.2. Vokallerin uzunluğu ve kısalığı

Vokallerin, oluşumlarındaki rahatlık yüzünden, diğer seslerden farklı bir tarafı da vardır. O da vokallerde uzunluk kısalık özelliklerinin bulunmasıdır. Vokaller açık bir geçitte rahat bir şekilde oluştukları için istenilen uzunlukta söylenebilirler. Onun için bazı dillerde bir vokalin uzunluk kısalık itibarıyla birden fazla şekli vardır.

1.1.5. Konsonant

Vokaller dışında kalan diğer seslere *konsonant* adı verilir. Konsonantların oluşumunda hava ve seda akımı ses geçidinde belirli bir takıntıya uğrar. Konsonantların bir kısmı sedalı, bir kısmı sedasızdır. Sedalı ve sedasız konsonantlar arasındaki fark

sadece bir seda farkı değil, aynı zamanda bir kuvvet farkıdır. Seda bakımından üç türlü konsonant vardır : Sedasız karşılıkları olan sedalı konsonantlar, sedasız karşılıkları olmayan sedalı konsonantlar, sedasız konsonantlar. Sedasız konsonantlar söylenirken genellikle bir aprisyon, sürtünmeden doğan *yelenme* meydana gelir. Fakat bu yelenme çok defa hissedilmez. Ancak, oluşumlarında organların temas ve yaklaşması çok olan bazı konsonantların sonunda duyurulur.

1.1.5.1. Konsonantların doğası ve özellikleri

Her dilde konsonantların sayısı vokallerinkinden çok fazladır. Fakat konsonantlar ses sınırlarını aşma yetenekleri bakımından, vokallerden daha zayıf bir durumda bulunmaktadır. Sesten başka sadece, ek olarak bir gramer birliği halinde görünebilirler. Tek başlarına, daha doğrusu vokalsiz kelime oluşturamazlar, kök olamazlar.

1.2. Türkçedeki sesler

1.2.1. Harf, alfabe, sesler

Sesler, adından da anlaşılacağı gibi, ağızdan çıkan, işitilen küçük dil birlikleridir. Dili yazıya geçirmek için bu küçük birlikler bir takım işaretlerle karşılanır. Bu işaretlere *harf* denir. Demek ki harf sesin yazıdaki işaretidir. Bir dildeki sesleri karşılayan harflerin hepsinin birden meydana getirdiği belli sıralı topluluğa *alfabe* denir.

Türkçedeki sesler genel alfabemizdeki işaretler ile ve sırasıyla şunlardır :

a b c ç d e f g ğ h ı i j k l m n o ö p r s
ş t u ü v y z

1.2.2. Vokaller

Türkçedeki bu seslerin içinde şu sekiz tanesi vokaldır :

a e ı i o ö u ü

Türkçe, vokal bakımından çok zengin bir dildir. Belli başlı dillerde genellikle 3 - 5 vokal bulunduğu halde Türkçede en aşağı 8 vokal vardır. En aşağı diyoruz, çünkü Türkçede özellikle konuşma dilinde açık e'nin yanında bir de e ile i arasında söylenen bir kapalı e vardır.

Türkçedeki vokallerden a vokali *at, taş, yayla* kelimelerindeki vokal; e vokali *et, sen, evet, dede* kelimelerindeki vokal; ı vokali *yıl, sıkı, kırıntı, ılık* kelimelerindeki vokal; i vokali *diş, iki, ilik, bilgin* kelimelerindeki vokal; o vokali *o, yol, ot, çok* kelimelerindeki vokal; ö vokali *ön, söz, göç, çöl* kelimelerindeki vokal; u vokali *su, uçuş, burun, uyku* kelimelerindeki vokal; ü vokali *üç, türk, gülünç, yüzük* kelimelerindeki vokaldır.

1.2.2.1. Vokallerin sınıflandırılması

İşte her birine bir kaç örnek vererek renklerini belittiğimiz Türkçedeki vokaller çeşitli bakımlardan gurup gurup birbirlerine yakınlık gösterirler. Vokallerin ayrımı bu yakınlıklara dayanır ve vokaller oluşum noktalarına göre, açıklık kapalılık derecelerine göre, oluşumları sırasında dudakların aldığı duruma göre, uzunluk ve kısalıklarına göre sınıflara ayrılırlar.

1.2.2.1.1. Kalın vokal, ince vokal

Oluşum noktalarına göre vokaller *art* ve *ön vokaller* yahut *kalın* ve *ince vokaller* olmak üzere ikiye ayrılır. *Art vokaller* dil ve damağın arka tarafında, *ön vokaller* dil ve damağın ön tarafında oluşan vokallerdir. Türkçede daha çok kullanılan deyimleri ile art vokallere *kalın vokaller*, ön vokallere de *ince vokaller* adı verilir. Sekiz vokalimizin dördü art yahut kalın vokal, dördü ise ön yahut ince vokaldır. Art yahut kalın vokaller *a, ı, o, u*

'dur. *e, i, ö, ü* ise ön yahut ince vokallerdir. Bunlardan en arkada oluşan vokal *u*, en önde oluşan vokal *i* 'dir. Kalın vokallerin arkadan öne doğru oluşum noktaları şöyle sıralanır: *u, o, ı, a*. İnce vokaller ise ikişer ikişer ön damağın ön ve arka taraflarında oluşurlar. *i* ile *ü* 'nün oluşum noktaları önde, *e* ile *ö* 'nünkü onların arkasındadır.

Bunlardan başka Türkçedeki yabancı kelimelerde kullanılan bir *a* daha vardır. Bu *a* ince bir vokaldir, oluşum noktası ön damak bölgesinin arkasında ve *a* ile *e* arasındadır. Örneğin *dikkat* ve *hakikat* kelimelerinin son vokalleri böyledir. Onun içindir ki *dikkati* veya *hakikate* dediğimiz zaman *a* vokalinden sonra *i, e* gibi ince vokaller getirmekteyiz.

1.2.2.1.2. Geniş vokal, dar vokal

Açıklık kapalılık derecelerine göre vokaller yine ikiye ayrılırlar. Çenelerin birbirine az veya çok yaklaşması sonucunda dilin kabaran kısmı ile damak arasındaki açıklık dar veya geniş olur. Açıklık fazla iken oluşan vokallere *geniş vokaller*, açıklık az iken oluşan vokallere de *dar vokaller* denir. Türkçedeki sekiz vokalin dördü geniş, dördü dar vokaldir. Geniş vokaller *a, e, o, ö*; dar vokaller ise *ı, i, u, ü* 'dür. Bunların içinde en geniş vokal *a* vokalıdır.

1.2.2.1.3. Düz vokal, yuvarlak vokal

Oluşumları sırasında dudakların aldığı *u r u m a* göre de vokaller iki kısma ayrılır. Dudakların açıklığı bir kısmının oluşumunda düz, bir kısmının oluşumunda ise yuvarlak bir durum alır. Oluşumlarında dudaklar düz olan vokallere *düz vokaller*, yuvarlak olan vokallere *yuvarlak vokaller* denir. Türkçenin sekiz vokalinin dördü düz, dördü yuvarlak vokaldir. Düz vokaller *a, e, ı, i*; yuvarlak vokaller ise *o, ö, u, ü* 'dür. Bunların içinde en düz vokal *e*, en yuvarlak vokal *ü* 'dür. Yuvarlak vokaller daraldıkları oranda dudak büzülmesi ve yuvarlakması artar.

1.2.2.1.4. Uzun vokal, kısa vokal

Uzunluk kısalıklarına göre vokaller sınıflara ayrılmazlar. Her vokalin uzun ve kısa şekli olur. Vokallerin kısalık ve uzunluğu boğumlama süresinin kısa veya uzun olmasıdır. Bir vokalin oluşumunun başladığı zamanla bittiği zaman arasındaki süre onun kısalık ve uzunluğunu gösterir. Aynı vokal bir darbe halinde çıkarılırsa onun kısa şekli, sürekli çıkarılırsa onun uzun şekli elde edilmiş olur. Bir vokalin uzunluk bakımından normal, normalden uzun ve normalden kısa olmak üzere üç şekli olabilir. Normal şekil vokallerin darbe halinde söylendikleri zaman gösterdikleri ses kulakta canlandırır, geçer. Yani bir süreklilik, tek sesinkinden fazla bir uzunluk duyulmaz. Vokallerin bu şekillerini uzun şekillerden ayırmak için böyle normal uzunluktaki vokallere *kısa vokaller* diyoruz. Normalden uzun olan, darbe ile değil süreklilikle çıkarılan vokallere de *uzun vokaller* adını veriyoruz. Vokallerin bu kısa ve uzun şekillerinden başka bir de normal bir ses uzunluğunu bile taşımayan normalden kısa şekilleri vardır. İşte Türkçede bu üç çeşit vokalden kısa vokal ile normal kısa vokal vardır. Uzun vokal Türkçede yoktur. Türkçedeki normal vokaller *a, e, i, o, ö, u, ü* 'dür. *ı* vokali ise normalden daha kısadır.

1.2.2.1.5. Sonuç

Yukarıdan beri söylediklerimizi şöyle toparlayabiliriz : Demek ki, hepsi kısa olan Türkçedeki sekiz vokalden *a, ı, o, u* kalın; *e, i, ö, ü* ince vokallerdir. Bunlardan *a, e, ı, i* düz; *o, ö, u, ü* yuvarlak; aynı zamanda *a, e, o, ö* geniş; *ı, i, u, ü* dar vokallerdir. Yani *a, e* düz geniş; *ı, i* dar düz; *o, ö* geniş yuvarlak; *u, ü* dar yuvarlak vokallerdir.

Vokaller oluşum noktaları, açıklık kapalılık, düzlük yuvarlaklık ve uzunluk kısalık bakımlarından başka, kuvvetlerine ve nazal olup olmadıklarına göre de sınıflara ayrılırlar. Fakat bu iki konu Türkçedeki vokalleri ilgilendirmemektedir. Çünkü Türkçedeki vokallerin söylenişinde bir kuvvet farkı yoktur. Aynı şekilde

Türkçede geniz vokali de yoktur.

1.2.3. Konsonantlar

Türkçedeki seslerin biraz önce gözden geçirdiğimiz sekiz vokal dışında kalanları, yani şu sesler konsonantlardır :

b c ç d f g ğ h j k l m n p r s ş t v y z

Bu sesler tek başlarına söylenirken sonlarına *ğ* 'da *ı* vokali, *k* 'da *a* vokali, diğerlerinde de bir *e* vokali getirilir.

Türkçedeki konsonantların karakterleri, renkleri şu misallerdeki söylenişleri gibidir :

b : boş, ebe; *bol*; *c* : gece, sıcak, acı; *ç* : çok, geçit, aç; *d* : dağ, dil, oda; *f* : fısıltı, üflemek, ufak; *g* : göz, bilgi, geniş; *ğ* : dağ, ağrı, oğul; *h* : hey, hangi, hele; *k* : küçük, iki, bekçi; *l* : al, el, gönül; *m* : yem, damla, gümüş; *n* : ben, anne, inanmak; *p* : tepe, parlak, köprü; *r* : duru, yaprak, demir; *s* : sevgi, ıslak, üst; *ş* : taş, şırıltı, düşüş; *t* : türk, atış, geçit; *v* : var, sivri, ova; *y* : ay, yeşil, oyun; *z* : yaz, güzel, kızgın.

j ise Türkçede asıllı olamayan ve Türkçedeki bir kaç yabancı kelimede kullanılan bir sestir : *jandarma*, *müşde*, *müşgan*, *jale*, *japon* gibi.

ğ harfi ile de bugün iki konsonant işaret edilmektedir. Bunlardan biri *bağ*, *boğum*, *yığın*, *ağlamak* gibi kelimelerde bulunan arka damak konsonantı *ğ* , diğeri de *geldiği*, *görmeğe*, *direğin* gibi kelimelerde bulunan ve *g* ile *k* 'nin iki vokal arasında yumuşamasından meydana geip *y* sesi veren fakat *ğ* harfi ile gösterilen konsonanttır. *yumuşak g* adı verilen ve *y* 'den farksız şekilde söylenen bu konsonantın *y* ile karıştırılmaması ve *g* ile *k* 'nin yumuşamasından türetildiğinin belirtilmesi içindir.

1.2.3.1. Konsonantların sınıflandırılması

Türkçede konsonantlar seda, oluşum noktası ve değme derecesi bakımından şu durumdadırlar :

1.2.3.1.1. Sedalı konsonant, sedasız konsonant

S e d a b a k ı m ı n d a n Türkçedeki konsonantlar sedalı ve sedasız olmak üzere ikiye ayrılırlar. Sedalı konsonantlar şunlardır: *b, c, d, g, ğ, j, l, m, n, r, v, y, z* . Sedasız konsonantlar da şunlardır: *ç, f, h, k, p, s, ş, t* . Sedasız konsonantların sedalıları arasında birer karşılıkları vardır. Bu karşılıklı durum onların oluşum noktalarının aynı yerde olmasındandır. Sedalı sedasız birbirinin karşılığı olan konsonantlar şunlardır : *b - p, c - ç, d - t, g - k, ğ - h, j - ş, v - f, z - s* . Sedasız karşılığı olmayan sedalı konsonantlar da *l* (ince ve kalın *l*), *m, n, r, y* sesleridir.

1.2.3.1.2. Oluşum noktalarına göre konsonantlar

O l u ş u m n o k t a s ı b a k ı m ı n d a n Türkçedeki konsonantlar çeşitli guruplara ayrılırlar. Bu gurupların oluşum noktaları gırtlaktan dudaklara kadar boğaz ve ağız yolunun çeşitli noktalarında bulunur. Bu noktalarda başta dil olmak üzere hareket eden uzuvlar birbirlerine yaklaşır veya dokunur. Önden arkaya doğru konsonantları oluşum noktalarına göre şöyle sıralayabiliriz :

Dudak konsonantları : *b, p, m* . Bunlar iki dudağın değmesi ile dudaklarda oluşan konsonantlardır.

Diş-dudak konsonantlar : *f, v* . Bunlar alt dudağın üst ön dişlere değmesi ile oluşan konsonantlardır.

Diş konsonantları : *d, t, n, s, z* . Bunlar dilin ucunun veya ön tarafının üst ön dişlerinin arkasına veya diş yuvalarına değmesi veya yaklaşması ile oluşan konsonantlardır. *t, d, n* 'de yolu kapatacak şekilde bir değme; *s, z* 'de ortada açık bir yol kalacak

şekilde bir yaklaşma olur. *n* 'de geniz yolu açık tutulur.

Damak - diş konsonantları : *c, ç, j, ş* . Bunlar dilin ucunun ve ön tarafının diş yuvası veya sert damağa değmesi veya yaklaşması ile oluşan konsonantlardır. *c, ç* 'de yolu kapatan bir temas olup oluşum noktası biraz önde; *j, ş* 'de yaklaşma olup oluşum noktası biraz arkadadır.

Ön damak konsonantları : *g, k, l, r, y* . Bunlar dilin ucunun veya orta tarafının sert yani ön damağa değmesi veya yaklaşması ile oluşan konsonantlardır. *g, k, y* 'de dilin orta tarafı kabarır; *r* 'de ise dilin ucu kalkar. *l* ince ve kalın olmak üzere iki tanedir. İnce *l* 'de dilin ön tarafı arka ince vokallerdeki gibi kabarır ve ucu kalkar, Kalın *l* 'de ise dilin arka tarafı arka vokallerdeki gibi kabarır ve ön tarafı dokunma için kalkar.

Arka damak konsonantı : *ğ* . Bu dilin arka tarafının yumuşak damağa değmesi veya yaklaşması ile oluşan konsonanttır. Kuvvetli bir sürtünmeye sebep olan bir yaklaşma olur. Bu konsonant arka damağın boğaza yakın noktalarında oluştuğu için buna arka damak - boğaz konsonantı da denilebilir.

Gırtlak konsonantı : *h* . Bu konsonant gırtlaktan önce hançerenin sonunda oluşur. Oluşumu sırasında ses telleri yarı açık bulunur.

1.2.3.1.3. Sürekli konsonantlar, süreksiz konsonantlar

Değme derecesi bakımından konsonantlar şu üç kısma ayrılırlar :

Değme derecesi tam olanlar : *b, c, ç, d, g, k, p, t* . Bunların oluşumu sırasında değen uzuvlar yolu tamamıyla kapatarak hava için hiç bir geçit bırakmazlar. Hava akımı önüne çıkan engele çarparak onu bir darbe ile açar ve konsonantın oluşmasına sebep olur. Bu konsonantlarda havanın, önündeki kapantıya çarpması ve onun açması ile bir patlama duyulur. Bu patlama bu konsonantın belirli

özelliklerini oluşturur. Onun için bu konsonantlara patlayıcı konsonantlar da denir. Patlama bu konsonantların sedalılarında hafif olur ve konsonantın ses tellerinden gelen sedasına karışır. Fakat sedasızlarda kuvvetli bir şekilde meydana gelerek açık şekilde işitilir. Demek ki değmenin derecesi ve hava akımı sedalılarda zayıf, sedasızlarda kuvvetlidir. Bu konsonantlar bir patlayışla, bir darbe ile oluştukları için, havanın önüne tam bir kapantı dikildiği için ancak bir darbe şeklinde söylenebilirler; sürekli bir şekilde çıkarılamazlar. Onun için bunlara *süreksiz konsonantlar* da denir.

Oluşumlarında hava için dar bir geçit bulunan konsonantlar : *f, ğ, h, j, s, ş, v, z* . Bunların oluşumunda hava değen uzuvlar arasındaki dar yoldan güçlkle geçer. Konsonantlar bu geçiş sırasında olan sürtünme ile oluşurlar.

Oluşumlarında hava için geniş bir yol bulunan konsonantlardır : *l* (kalın ve ince), *m, n, r, y* . Bunların oluşumunda değen veya yaklaşan uzuvlar arasında hava için rahat bir yol kalır. Değme derecesi en az olan konsonantlar bunlardır. Bunlardan daha az değme derecesine sahip olan sesler ise vokallerdir. Değme derecelerinin vokallere yakınlığı dolayısıyla bu konsonantlara vokal - konsonant adı da verilir. İçlerinde vokallere en yakın olanı *y* 'dir. *y* 'nin oluşumunda uzuvların durumu *i* 'ninkine çok yaklaşır.

Oluşumlarında hava için dar bir geçit bulunan konsonantlarla geniş bir geçit bulunan konsonantlar sürekli bir şekilde çıkarılabilirler. Bu yüzden bu iki değme dercesindeki konsonantlara *sürekli konsonantlar* adı verilir. —

1.2.3.1.4. Geniz konsonantı, ağız konsonantı

Seda, oluşum noktası ve değme dercesi bakımından yaptığımız bu sınıflamalardan başka konsonantlar bir de *n a z a l o l u p o l m a m a b a k ı m ı n d a n* ikiye ayrılırlar. Türkçede iki nazal konsonant vardır. Onlar da *m, n* 'dir. Bunların oluşumunda hava

geniz yolundan çıkar. Neze olunca insanın bu sesleri iyi söyleyemesinin sebebi de budur. Bunların dışında kalan diğer konsonantlar, ağız konsonantları olup oluşumlarında geniz yolu kapalı tutulur.

BÖLÜM 2

2. Macintosh ve MacinTeik

2.1. Macintosh

2.1.1. Macintosh Plus 'un teknik özellikleri

İşlemci : MC 68000, 32 bit iç yapı, saat 7.8336 MHz

Bellek : 1 MB RAM

128 KB ROM

Kullanıcı tarafından değiştirilebilir

parametreler için pille desteklenen 256 byte

Disk Kapasitesi : 2 disk 5.25" 800K

Ara Birimler : Ek zamanla (sankronize) seri klavye yoldu

iki RS232C/RS422 seri çıkış, maksimum 230.4K

baud (herlei saat kullanıldığında

0.920 megabit'e kadar yükseltilebilir)

Mouse ara birimi

Hardisk ara birimi

SCSI ara birimi

BÖLÜM 2

2.1

Ses Üretimi : 22KHz örnekleme hızında 8-bit sayısalan
örnekleme (digital to analog) çevirimi
4 farklı frekansla ses.

2.1.2. Macintosh İşletim Sistemi

Macintosh İşletim Sistemi, 'Menu Oriented (Menüye Dayalı)' bir işletim sistemidir. Yani komutlar menüler aracılığı ile yerine getirilir. Bu komutları işletimye aktaran program ise 'Finder' dir. Finder, Macintosh'da dokümanları kullanabilmek için gerekli yönetilebilir sürümü programdır. Bilinen işletim sistemlerinin yerine getirildiği komutları Macintosh İşletim Sistemi'nin de yerine getirmek için Finder'da komutlar 'Mouse' yardımı ile seçilir.

2.2. MacinTalk

2. Macintosh ve MacinTalk

2.2.1. Genel Bilgi

2.1. Macintosh

MacinTalk, Macintosh Konuşma Sentezicisi, Macintosh İşletim Sistemi, Macintosh Plus 'un teknik özellikleri

İşlemci : MC 68000, 32-bit iç yapı, saat 7.8336 MHz

Bellek : 1 MB RAM

128 KB ROM

Reader, işlemi sürdüren kullanıcı tarafından değiştirilebilir bir menü
MacinTalk'un arayüzü parametreler için pille desteklenen 256 byte
çevirimi bellek

Disk Kapasitesi : 2 tane 3.5 inçlik çift yüzlü disket, 800K

1 tane SCIS sabit disk, 20MB

Ara Birimler : Eş zamanlı(senkronize) seri klavye yolu

İki RS232/RS422 seri çıkış, maksimum 230.4K
baud (harici saat kullanıldığında

0.920 megabit'e kadar yükseltilir)

Mouse ara birimi

Harici disk arabirimi

SCSI ara birimi

2 . 2

Ses Üretici : 22KHz örnekleme hızında 8-bit sayısal dan örneksele (digital to analog) çevrilmiş 4 farklı frekansta ses.

2.1.2. Macintosh İşletim Sistemi

Macintosh İşletim Sistemi, 'Menu Oriented (Menü'ye Dayalı)' bir işletim sistemidir. Yani komutlar menüler aracılığı ile yerine getirilir. Bu komutları işlemciye aktaran program ise '**Finder**' dir. Finder, Macintosh'da dokümanları kullanabilmek için trafiği yönetmekle sorumlu programdır. Bilinen işletim sistemlerinin yerine getirdiği komutları Macintosh İşletim Sistemi'n de yerine getirmek için Finder'de komutlar '**Mouse**' yardımı ile seçilir.

2.2. MacInTalk

2.2.1. Genel Bilgi

MacInTalk, Macintosh Konuşma Sentezcisi, Macintosh işletim sistemi altında çalışan bir yazılım sürücüsüdür. Bu sürücü fonetik kodların ASCII karşılıklarını yüksek seviyeli bireşimli konuşmaya çevirir. Konuşma süresi ve sesin alçalma ve yükselmeleri programlanabilir.

Reader isimli sürücünün bir bileşeni, standard bir metni **MacInTalk**'un anlayabileyeceği fonetik kodlara çevirir. Ancak bu çevirme %100 başarılı olmayabilir. Yapılan deneylere göre, %70-%80 arasında bu başarı oranı değişmektedir. Hata oluştuğunda bir insanın yaptığı bir hata gibi algılanmaktadır.

2.2.2. MacInTalk sürücüsünü kullanma

MacInTalk sürücüsünü kullanan herhangi bir yazılım uygulaması bazı standard alt programları kullanmalıdır. Birincisi, **SpeechOn**, sürücüyü ilk durumuna getirir ve bir dönüş kodu ile geri döner. Daha çok kullanılan alt program **MacInTalk**, fonetik koda dönüştürülmüş

2.3

yazıyı normal konuşma yapmak için kullanılır. Sonunda sürücüyü kapatmak için **SpeechOff** alt programı kullanılır.

Bu fonksiyonların yanında konuşmanın hızını belirleyen **SpeechRate**, sesin şiddetini belirleyen **SpeechPitch** alt programları da vardır.

**FUNCTION MacinTalk(theSpeech : SpeechHandle;
Phonemes : Handle) : SpeechErr;**

Phonemes karakter dizisinin fonetik karşılığı olan bir dizidir. Bu dizi büyük harf olmalı ve sadece fonetik kodlardan oluşmalıdır.

Olabilir dönüş (hata) değerleri :

noError : No error

nilHandleErr : Giriş tutanağı veya esas pointer NIL dir.

positive integer : Giriş dizisinde hata vardır ve bu sayı hatalı yerin dizi içinde sondan kaçınıcı olduğunu belirtir.

**FUNCTION Reader (theSpeech : SpeechHandle;
EnglishPtr : Ptr; InputLength : LONGINT;
PhoneticOutput : Handle) : SpeechErr;**

Reader fonksiyonu metni fonetik forma dönüştürür. EnglishPtr metnin bulunduğu diziyi gösteren pointerdir. InputLength bu dizinin uzunluğudur. PhoneticOutput fonetik karakterlerin bulunduğu diziyi belirler.

3. Program hakkında bilgi

3.1. Genel bilgi

İlk önce ses sürücüsü açılır. Türkçe metin 'İçerik' adlı file'dan bir karakter dizisine okunur. Sonra metin içindeki bazı Türkçe karakterler İngilizce karakter karşılıkları oluşturulur. Sonra eide edilen bu dizinin fonetik forma dönüştürülür.

Eide edilmiş olan bu fonetik formdaki fonetik karakterler kullanılarak metnin konusuna karşılığı eide edilir. Daha sonra sürücü kapanarak program sona erdirilir.

BÖLÜM 3

3.2. Türkçe metnin hazırlanması

İstenilen herhangi Türkçe metnin bir Editor yardımı ile diske kaydedilir. Bu metin yazılırken Türkçe karakterler kullanılır. Bunun içinde Macintosh'ın klavyesinde bulunan 'Option' tuşu kullanılır. Tabii bu durum 'q' tipi klavye için geçerlidir. Eğer 'F' tipi klavye kullanılıyorsa bu iş için 'Option' tuşuna gerek kalmaz. Türkçe

3.1

alfabede bulunan 2, 3, 4, 5, 6, 7 ve 8, 9, 0, 1 karakterleri. Option tuşu ile bu harflerin klavyede bulunan noktasızlarına beraber bastırarak elde edilir.

Editor olarakta Edit, MacWrite gibi programlar kullanılabilir. Türkçe metnin bulunduğu file'in adı mutlaka 'İçerik' olmalıdır.

3.2. 'Konuş' adlı programın geliştirilmesi

Konuş adlı program mouse yardımı ile açılır ve geliştirilmesi sağlanır. Daha önce yazılmış olan Türkçe metin Macintosh'un harddiskinden duyulur.

3. Program hakkında bilgi

3.1. Genel bilgi

İlk önce ses sürücüsü açılır. Türkçe metin 'İçerik' adlı file'dan bir karakter dizisine okunur. Sonra metin içindeki bazı Türkçe karakterler İngilizce karakter karşılıkları oluşturulur. Sonra elde edilen bu dizi fonetik forma dönüştürülür.

Elde edilmiş olan bu fonetik formdaki fonetik karakterler kullanılarak metnin konuşma karşılığı elde edilir. Daha sonra sürücü kapanarak program sona erdirilir.

3.2. Türkçe metnin hazırlanması

İstenilen herhangi Türkçe metin bir Editor yardımı ile diske kaydedilir. Bu metin yazılırken Türkçe karakterler kullanılır. Bunun içinde Macintosh'un klavyesinde bulunan 'Option' tuşu kullanılır. Tabii bu durum q tipi klavye için geçerlidir. Eğer F tipi klavye kullanılıyorsa bu iş için Option tuşuna gerek kalmaz. Türkçe

3.2

alfabede bulunan , ę, , , , ı ve , , , , , ı karakterleri, Option tuşu ile bu harflerin klavyede bulunan noktasızlarına beraber basılarak elde edilir.

Editor olarakda **Edit, MacWrite** gibi programlar kullanılabilir. Türke metnin bulunduęu file'in adı mutlaka '**İerik**' olmalıdır.

3.3. 'Konuş' adlı programın alıştırılması

Konuş adlı program mouse yardımı ile seçilir ve alıştırılması sağlanır. Daha önce yazılmış olan Türke metin Macintosh'un hoparlöründen duyulur.

PROGRAM KONUSU

Program adı: KONUS

Fonksiyon: Verilen Türkçe metni konuşturma

Yazar: H.Cemal Terzioğlu

(Si MemTypes.pas)

(Si QuickDraw.pas)

(Si OSInt.pas)

(Si ToolInt.pas)

(Si Speech.pas)

TYPE

CharArray = packed array [1..120] of char;

CharArrayPtr = ^CharArray;



VAR

numChars: integer;

SErr: SpeechErr;

TextIn: CharArray; { Tüğe metni }

Meta: CharArray;

MetaPtr: CharArrayPtr; { meta'nın pointer'i }

TextFile: Text; { Giris file'i }

Phonemes: Handle; { Fonetiklerin tutanağı }

theSpeech: SpeechHandle; { Konusma tutanağı }

BEGIN

{ Sörsöyü açma }

SErr := SpeechGet theSpeech;

if SErr = noErr then begin

Open(TextFile, fank);

Read(TextFile);

numChars := 1;

PROGRAMLAR


```

PROGRAM Konus;
{
  Program adı : KONUS
  Fonsiyonu   : Varolan Türkçe metni konuşurma
  Yazan       : H.Cahit Terzioğlu
}

```

```

{$i MemTypes.ipas }
{$i QuickDraw.ipas }
{$i OSIntf.ipas   }
{$i ToolIntf.ipas }
{$i Speech.ipas   }

```

TYPE

```

CharArray = packed array[1..5120] of char;
CharArrayPtr = ^ CharArray;

```

VAR

```

numChars,i,j : Integer;
SErr: SpeechErr;
TextIn: CharArray; { Türkçe metin }
Metin: CharArray;
MetinPtr: CharArrayPtr; { metnin pointer'i }
TextFile: Text; { Giriş file'ı }
Phonetics: Handle; { Fonetiklerin tutanağı }
theSpeech: SpeechHandle; { Konuşma tutanağı }

```

BEGIN

```

{ Sürücüyü açma }
SErr := SpeechOn(",theSpeech);

```

```

if SErr = noErr then begin
  Open(TextFile,'İçerik');
  Reset(TextFile);

```

```

  numChars := 1;

```



```

PROGRAM Konus;
{
  Program adı : KONUS
  Fonsiyonu   : Varolan Türkçe metni konuşurma
  Yazan       : H.Cahit Terzioğlu
}

```

```

{$i MemTypes.ipas }
{$i QuickDraw.ipas }
{$i OSIntf.ipas   }
{$i ToolIntf.ipas }
{$i Speech.ipas   }

```

TYPE

```

  CharArray   = packed array[1..5120] of char;
  CharArrayPtr = ^ CharArray;

```

VAR

```

  numChars,i,j : Integer;
  SErr: SpeechErr;
  TextIn: CharArray; { Türkçe metin }
  Metin: CharArray;
  MetinPtr: CharArrayPtr; { metnin pointer'i }
  TextFile: Text; { Giriş file'ı }
  Phonetics: Handle; { Fonetiklerin tutanağı }
  theSpeech: SpeechHandle; { Konuşma tutanağı }

```

BEGIN

```

  { Sürücüyü açma }
  SErr := SpeechOn("",theSpeech);

```

```

  if SErr = noErr then begin
    Open(TextFile,'İçerik');
    Reset(TextFile);

```

```

    numChars := 1;

```



```

while not eof(TextFile) do begin
    TextIn[numChars] := TextFile^;
    numChars := numChars + 1;
    get(TextFile);
end;

```

```

{ Türkçe karakterlerin karşılıklarının bulunması }

```

```

j := 1;
for i := 1 to numChars - 1 do
begin
    j := j + 1;
    case TextIn[i] of
        'c' : begin
            Metin[j] := 'd';
            j := j + 1;
            Metin[j] := 'j';
        end;
        'ç' : begin
            Metin[j] := 'c';
            j := j + 1;
            Metin[j] := 'h';
        end;
        'ğ' : begin
            Metin[j] := 'g';
            j := j + 1;
            Metin[j] := 'h';
        end;
        'ı' : Metin[j] := 'i';
        'o' : begin
            Metin[j] := 'o';
            j := j + 1;
            Metin[j] := 'w';
        end;
        'ü' : begin
            Metin[j] := 'u';
            j := j + 1;

```



```
        Metin[j]:='e';
    end;
'u' : begin
        Metin[j]:='u';
        j := j + 1;
        Metin[j]:='h';
    end;
's' : begin
        Metin[j]:='s';
        j := j + 1;
        Metin[j]:='h';
    end;
'ö' : begin
        Metin[j]:='o';
        j := j + 1;
        Metin[j]:='e';
    end;
'1' : begin
        Metin[j]:='b';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='r';
    end;
'2' : begin
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='k';
        j := j + 1;
        Metin[j]:='i';
    end;
```



```
'3' : begin
    Metin[j]:='u';
    j := j + 1;
    Metin[j]:='e';
    j := j + 1;
    Metin[j]:='c';
    j := j + 1;
    Metin[j]:='h';
end;
```

```
'4' : begin
    Metin[j]:='d';
    j := j + 1;
    Metin[j]:='o';
    j := j + 1;
    Metin[j]:='e';
    j := j + 1;
    Metin[j]:='r';
    j := j + 1;
    Metin[j]:='t';
end;
```

```
'5' : begin
    Metin[j]:='b';
    j := j + 1;
    Metin[j]:='e';
    j := j + 1;
    Metin[j]:='s';
    j := j + 1;
    Metin[j]:='h';
end;
```

```
'6' : begin
    Metin[j]:='a';
    j := j + 1;
    Metin[j]:='l';
    j := j + 1;
    Metin[j]:='t';
    j := j + 1;
```



```

        Metin[j]:='i';
    end;
'7' : begin
        Metin[j]:='y';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='d';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='e';
    end;
'8' : begin
        Metin[j]:='s';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='k';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='e';
        j := j + 1;
        Metin[j]:='z';
    end;
'9' : begin
        Metin[j]:='d';
        j := j + 1;
        Metin[j]:='o';
        j := j + 1;
        Metin[j]:='k';
        j := j + 1;
        Metin[j]:='o';
        j := j + 1;
        Metin[j]:='o';
    end;

```



```

        j := j + 1;
        Metin[j] := 'z';
    end;
'0' : begin
        Metin[j] := 's';
        j := j + 1;
        Metin[j] := 'i';
        j := j + 1;
        Metin[j] := 'f';
        j := j + 1;
        Metin[j] := 'i';
        j := j + 1;
        Metin[j] := 'r';
    end;
otherwise Metin[j] := TextIn[i];
end
end;

MetinPtr := @Metin;

Phonetics := NewHandle(0);

{ Text'i fonetik kodlara dönüştürme }
SErr := Reader(theSpeech, Pointer(MetinPtr), numChars,
               Phonetics);

{ Dönüştürülmüş text'i konuşurma }
SErr := MacinTalk(theSpeech, Phonetics);

{ Sürücüyü kapatma }
SpeechOff(theSpeech);

DisposHandle(Phonetics);

Close(TextFile)
end

```


end.

MENTYPES Pascal/MacIIcs MENTYPES

NAME

MemTypes - common data types (16 Jan 85)

DESCRIPTION

These types are shared by several Macintosh libraries.

AUTHOR

Pascal Interfaces by Ronny Sabok
Copyright 1985, 1984, 1985 Apple Computer Inc.

MODIFICATIONS

Tom Leppard, DML Systems

Changed to "include file" format for use with MacLanguage
Series Pascal

TYPE

SignedByte = 128..127; { any byte in memory }

Byte = 0..255; { unsigned byte for fontmgr }

Ptr = SignedByte; { fixed pointer }

Handle = ^Ptr; { pointer to a master pointer }

ProcPtr = Ptr; { pointer to a procedure }

Fixed = LongInt; { fixed point arithmetic type }

Str255 = String[255]; { maximum string size }

StringPtr = ^Str255; { pointer to maximum string }

StringHandle = ^StringPtr; { handle to maximum string }

{-----}

MEMTYPES Pascal Interface MEMTYPES

NAME

MemTypes -- common data types (16 Jan 85)

DESCRIPTION

These types are shared by several Macintosh libraries. System

AUTHOR

Pascal Interfaces by Rony Sebok
Copyright 1983, 1984, 1985 Apple Computer Inc.

MODIFICATIONS

Tom Leonard, TML Systems

Changed to "include file" format for use with MacLanguage
Series Pascal

-----}

TYPE

SignedByte= -128..127; { any byte in memory }

Byte= 0..255; { unsigned byte for fontmgr }

Ptr= ^ SignedByte; { blind pointer }

Handle= ^ Ptr; { pointer to a master pointer }

ProcPtr= Ptr; { pointer to a procedure }

Fixed= LongInt; { fixed point arithmetic type }

Str255= String[255]; { maximum string size }

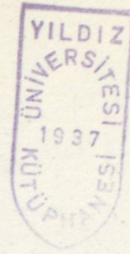
StringPtr= ^ Str255; { pointer to maximum string }

StringHandle= ^ StringPtr; { handle to maximum string }

MODIFICATIONS

Tom Leonard, TML Systems

Changed to "include file" format for use with MacLanguage



{-----}

OSINTF Pascal Interface OSINTF February 1985

NAME

OSIntf -- Operating System Interface (8 Feb 85)

DESCRIPTION OSTrap Rel is provide for EXTERNAL subprograms

This file contains the interface to Macintosh Operating System routines.

Most of these routines required some assembly language interface code.

For more detailed information see the following sections of the "Inside Macintosh" documentation:

- Device Manager
- Disk Driver
- File Manager
- Memory Manager
- OS Event Manager
- OS Utilities
- Segment Loader
- Serial Driver
- Sound Driver
- System Error
- Vertical Retrace Manager

AUTHOR

Pascal Interfaces by Rony Sebok
Copyright 1983, 1984, 1985 Apple Computer Inc.

MODIFICATIONS

Tom Leonard, TML Systems
Changed to "include file" format for use with MacLanguage

Series Pascal

Additions for the 128K ROMs are added February 1986.
DO NOT attempt to use these with the 64K ROMs.

----- }

{ Link the file OSTraps.Rel to provide for EXTERNAL subprograms }
{ \$u OSTraps }

CONST {for Event Manager}

everyEvent = -1;
NullEvent = 0;
mouseDown = 1;
mouseUp = 2;
keyDown = 3;
keyUp = 4;
autoKey = 5;
updateEvt = 6;
diskEvt = 7;
activateEvt = 8;
networkEvt = 10;
driverEvt = 11;
app1Evt = 12;
app2Evt = 13;
app3Evt = 14;
app4Evt = 15;

{ event mask equates }

mDownMask = 2;
mUpMask = 4;
keyDownMask = 8;
keyUpMask = 16;
autoKeyMask = 32;
updateMask = 64;

Pr 3 - 3

```
diskMask      = 128;
activMask     = 256;
networkMask   = 1024;
driverMask    = 2048;
app1Mask      = 4096;
app2Mask      = 8192;
app3Mask      = 16384;
app4Mask      = -32768;
```

```
{to decipher event message for keyDown events}
```

```
charCodeMask = $000000FF;
```

```
keyCodeMask  = $0000FF00;
```

```
{ modifiers }
```

```
optionKey= 2048; { Bit 3 of high byte }
```

```
alphaLock= 1024; { Bit 2 }
```

```
ShiftKey = 512; { Bit 1 }
```

```
CmdKey   = 256; { Bit 0 }
```

```
BtnState = 128; { Bit 7 of low byte is mouse button state }
```

```
activeFlag = 1; { bit 0 of modifiers for activate event }
```

```
{error for PostEvent}
```

```
EvtNotEnb = 1;
```

```
{for Memory Manager}
```

```
MemFullErr = -108; { Not enough room in heap zone }
```

```
NilHandleErr = -109; { Master Pointer was NIL in HandleZone or  
other }
```

```
MemWZErr    = -111; { WhichZone failed (applied to free block) }
```

```
MemPurErr    = -112; { trying to purge a locked or non-purgeable  
block }
```

```
MemLockedErr = -117; { Block is locked }
```

```
NoErr        = 0; { All is well }
```



```

{file system error codes}
DirFulErr  =   -33; { Directory full}
DskFulErr  =   -34; { disk full}
NSVErr     =   -35; { no such volume}
IOErr      =   -36; { I/O error (bummers)}
BdNamErr   =   -37; { there may be no bad names in the final
system!}
FNOpnErr   =   -38; { File not open}
EOFErr     =   -39; { End of file}
PosErr     =   -40; { tried to position to before start of file
(r/w)}
MFulErr    =   -41; { memory full(open) or file won't fit (load)}
TMFOErr    =   -42; { too many files open}
FNFErr     =   -43; { File not found}

WPrErr     =   -44; { diskette is write protected}
FLckdErr   =   -45; { file is locked}
VLckdErr   =   -46; { volume is locked}
FBsyErr    =   -47; { File is busy (delete)}
DupFNErr   =   -48; { duplicate filename (rename)}
OpWrErr     =   -49; { file already open with with write
permission}
ParamErr    =   -50; { error in user parameter list}
RFNumErr    =   -51; { refnum error}
GFPErr     =   -52; { get file position error}
VolOffLinErr=   -53; { volume not on line error (was Ejected)}
PermErr     =   -54; { permissions error (on file open)}
VolOnLinErr =   -55; { drive volume already on-line at MountVol}
NSDrvErr    =   -56; { no such drive (tried to mount a bad
drivenum)}
NoMacDskErr =   -57; { not a mac diskette (sig bytes are wrong)}
ExtFSErr    =   -58; { volume in question belongs to an external
fs}
FSRnErr     =   -59; { file system rename error;}
BadMDBErr   =   -60; { bad master directory block}
WrPermErr   =   -61; { write permissions error}

```



```

lastDskErr  = -64; { last of the range of low-level disk errors}
noDriveErr  = -64; { drive not installed }
offLinErr   = -65; { r/w requested for an off-line drive }
noNybErr    = -66; { couldn't find 5 nybbles in 200 tries }
noAdrMkErr  = -67; { couldn't find valid addr mark }
dataVerErr  = -68; { read verify compare failed }
badCkSmErr  = -69; { addr mark checksum didn't check }
badBtSlpErr = -70; { bad addr mark bit slip nibbles }
noDtaMkErr  = -71; { couldn't find a data mark header }
badDCkSum   = -72; { bad data mark checksum }
badDBtSlp   = -73; { bad data mark bit slip nibbles }
wrUnderRun  = -74; { write underrun occurred }
cantStepErr = -75; { step handshake failed }
tk0BadErr   = -76; { track 0 detect doesn't change }
initIWMErr  = -77; { unable to initialize IWM }
twoSideErr  = -78; { tried to read 2nd side on a 1-sided drive }
spdAdjErr   = -79; { unable to correctly adjust disk speed }
seekErr     = -80; { track number wrong on address mark }
sectNFErr   = -81; { sector number never found on a track }
firstDskErr = -84; { first of the range of low-level disk
errors}

```

```

DirNFErr    = -120; { Directory not found }
TMWDOErr    = -121; { No free WDCB available }
BadMovErr    = -122; { Move into offspring error }
WrgVolTypErr= -123; { Wrong volume type error - operation not
supported for MFS}
FSDSIntErr  = -127; { Internal file system error }

```

```

MaxSize     = $800000; { Max data block size is 512K bytes }

```

```

{finder constants}

```

```

fHasBundle  = 8192;

```

```

fInvisible  = 16384;

```

```

fTrash      = -3;

```


Pr 3 - 6

```
fDesktop = -2;  
fDisk    = 0;
```

{io constants}

{ioPosMode values}

```
fsAtMark    = 0;  
fsFromStart = 1;  
fsFromLEOF  = 2;  
fsFromMark  = 3;  
rdVerify    = 64;
```

{ioPermission values}

```
fsCurPerm   = 0;  
fsRdPerm     = 1;  
fsWrPerm     = 2;  
fsRdWrPerm   = 3;  
fsRdWrShPerm = 4;
```

{refNums from the serial ports}

```
AinRefNum = -6;    {serial port A input}  
AoutRefNum = -7;   {serial port A output}  
BinRefNum  = -8;   {serial port B input}  
BoutRefNum = -9;   {serial port B output}
```

{baud rate constants}

```
baud300 = 380;  
baud600 = 189;  
baud1200 = 94;  
baud1800 = 62;  
baud2400 = 46;  
baud3600 = 30;  
baud4800 = 22;  
baud7200 = 14;  
baud9600 = 10;  
baud19200 = 4;
```


baud57600= 0;

{SCC channel configuration word}

{driver reset information masks}

stop10 = 16384;

stop15 = -32768;

stop20 = -16384;

noParity = 0;

oddParity = 4096;

evenParity = 12288;

data5 = 0;

data6 = 2048;

data7 = 1024;

data8 = 3072;

{serial driver error masks}

swOverrunErr = 0;

parityErr = 16;

hwOverrunErr = 32;

framingErr = 64;

{serial Port configuration usage constants for Config}

{field of SysParmType}

useFree = 0;

useATalk = 1;

useAsync = 2;

{serial driver message constant}

xOffWasSent = \$80;

{for application parameter}

{constants for message returned by the finder on launch}

appOpen = 0;

appPrint = 1;

{for sound driver}

SWmode = -1;

FTmode = 1;

FFmode = 0;

{for "machine" parameter of Environs}

macXMLMachine = 0;

macMachine = 1; { reserved for future }

ioDirFlg = 3; { IF BitTst(ioDirFlg, myParamBlk^.ioFIAttrib)
THEN ... }

ioDirMask = \$10; { IF BitAnd(ioDirMask,
myParamBlk^.ioFIAttrib) = ioDirMask THEN ... }

FSRtParID = 1; { DirID of parent's root }

FSRtDirID = 2; { Root DirID }

TYPE {for Event Manager} generic queue element

EventRecord =

RECORD Marking Control Block Queue Element

what: Integer;

message: LongInt;

when: LongInt; Ptr; {link to next element}

where: Point; {unique ID for validity check}

modifiers: Integer; {address of service routine}

END; {integer; {count field for timeout}

{integer; {phase to allow synchronization}

Zone =

RECORD

BkLim: Ptr; {Bk Task}

PurgePtr: Ptr;

HFstFree: Ptr;

ZCBFree: LongInt;

GZProc: ProcPtr;

MoreMast: Integer;

Flags: Integer; {this part is identical to the

CntRel: Integer; {EventRecord defined in Toolint}

MaxRel: Integer;


```

CntNRel: Integer;
MaxNRel: Integer;
CntEmpty: Integer;
CntHandles: Integer;
MinCBFree: LongInt;
PurgeProc: ProcPtr;
SparePtr: Ptr; { reserved for future }
AllocPtr: Ptr;
HeapData: Integer;
END;

```

```

THz = ^Zone; { pointer to the start of a heap zone }
Size = LongInt; { size of a block in bytes }
OsErr = Integer; { error code }

```

```

QElemPtr = ^QElem; {ptr to generic queue element}

```

```

{Vertical Blanking Control Block Queue Element}

```

```

VBLTask =

```

```

    RECORD

```

```

        qLink: QElemPtr; {link to next element}
        qType: Integer; {unique ID for validity check}
        vblAddr: ProcPtr; {address of service routine}
        vblCount: Integer; {count field for timeout}
        vblPhase: Integer; {phase to allow synchronization}
    END;

```

```

{VBLQEIPtr = ^VBLTask;}

```

```

evQEI =

```

```

    RECORD

```

```

        qLink: QElemPtr;
        qType: Integer;
        evtQwhat: Integer; {this part is identical to the }
        evtQmessage: LongInt; {EventRecord defined in ToolIntf}
        evtQwhen: LongInt;

```



```

evtQwhere: Point;
evtQmodifiers: Integer;
END;

```

{drive queue elements}

```

DrvQEI =
  RECORD
    qLink: QElemPtr;
    qType: Integer;
    dQDrive: Integer;
    dQRefNum: Integer;
      {ref num of the drvr which handles this drive}
    dQFSID: Integer;
      {id of file system which handles this drive}
    dQDrvSize: Integer;
      {size of drive in 512-byte blocks -- not for drvs 1&2}
  END;

```

```

DrvQEIPtr = ^DrvQEI;

```

```

ParamBlkType = (IOParam, FileParam, VolumeParam, CntrlParam);

```

```

OsType = PACKED ARRAY[1..4] OF CHAR; {same as rsrc mgr's
Restype}

```

```

FInfo =
  RECORD {record of finder info}
    fdType: OsType; {the type of the file}
    fdCreator: OsType; {file's creator}
    fdFlags: Integer; {flags ex. hasbundle, invisible, locked, etc.}
    fdLocation: Point; {file's location in folder}
    fdFldr: Integer; {folder containing file}
  END;

```

```

FXInfo =

```


RECORD

FFIconID: INTEGER; {Icon ID}

FFUnused: ARRAY[1..4] OF INTEGER; {unused but reserved 8 bytes}

FFComment: INTEGER; {Comment ID}

FFPutAway: LONGINT; {Home Dir ID}

END;

DInfo =

RECORD

FRRect: Rect; {folder rect}

FRFlags: INTEGER; {Flags}

FRLocation: Point; {folder location}

FRView: INTEGER; {folder view}

END;

DXInfo =

RECORD

FRScroll: Point; {scroll position}

FROpenChain: LONGINT; {DirID chain of open folders}

FRUnused: INTEGER; {unused but reserved}

FRComment: INTEGER; {comment}

FRPutAway: LONGINT; {DirID}

END;

ParamBlockRec =

RECORD {12 byte header used by the file and IO system}

qLink: QElemPtr; {queue link in header}

qType: Integer; {type byte for safety check}

ioTrap: Integer; {FS: the Trap}

ioCmdAddr: Ptr; {FS: address to dispatch to}

{common header to all variants}

ioCompletion: ProcPtr;

{completion routine addr (0 for synch calls)}

ioResult: OsErr; {result code}

ioNamePtr: StringPtr; {ptr to Vol:FileName string}

ioVRefNum: Integer;

{volume refnum (DrvNum for Eject and MountVol)}

{different components for the different}

{type of parameter blocks}

CASE ParamBlkType OF

ioParam:

(ioRefNum: Integer; {refNum for I/O operation}

ioVersNum: SignedByte; {version number}

ioPermsn: SignedByte; {Open: permissions (byte)}

ioMisc: Ptr; {Rename: new name}

{GetEOF,SetEOF: logical end of file}

{Open: optional ptr to buffer}

{SetFileType: new type}

ioBuffer: Ptr; {data buffer Ptr}

ioReqCount: LongInt; {requested byte count}

ioActCount: LongInt; {actual byte count completed}

ioPosMode: Integer; {initial file positioning}

ioPosOffset: LongInt); {file position offset}

FileParam:

(ioFRefNum: Integer; {reference number}

ioFVersNum: SignedByte; {version number}

filler1: SignedByte;

ioFDirIndex: Integer; {GetFileInfo directory index}

ioFIAttrib: SignedByte; {GetFileInfo: in-use bit=7, lock bit=0}

ioFIVersNum: SignedByte; {file version number}

ioFIFndrInfo: FInfo; {user info}

ioFINum: LongInt; {GetFileInfo: file number}

ioFIStBlk: Integer; {start file block (0 if none)}

ioFILgLen: LongInt; {logical length (EOF)}

ioFIPyLen: LongInt; {physical length}

ioFIRStBlk: Integer; {start block rsrc fork}

ioFIRLgLen: LongInt; {file logical length rsrc fork}


```
ioFIRPyLen: LongInt; {file physical length rsrc fork}
ioFICrDat: LongInt; {file creation date & time (32 bits in
secs)}
ioFIMdDat: LongInt); {last modified date and time}
```

VolumeParam:

```
{filler2: LongInt;
ioVolIndex: Integer; {volume index number}
ioVCrDate: LongInt; {creation date and time}
ioVLsBkUp: LongInt; {last backup date and time}
ioVAtrb: Integer; {volume attrib}
ioVNmFls: Integer; {number of files in directory}
ioVDirSt: Integer; {start block of file directory}
ioVBILn: Integer; {GetVollInfo: length of dir in blocks}
ioVNmAlBlks: Integer; {GetVollInfo: num blks (of alloc size)}
ioVAIBlkSiz: LongInt; {GetVollInfo: alloc blk byte size}
ioVClpSiz: LongInt; {GetVollInfo: bytes to allocate at a time}
ioAIBlSt: Integer; {starting disk(512-byte) block in block
map}
ioVNxtFNum: LongInt; {GetVollInfo: next free file number}
ioVFrBlk: Integer); {GetVollInfo: # free alloc blks for this vol}
```

CntrlParam:

```
{ioCRefNum: Integer; {refNum for I/O operation}
CSCode: Integer; {word for control status code}
CSParm: ARRAY[0..10] OF Integer);{operation-defined
parameters}
END; {ParamBlockRec}
```

ParmBlkPtr = ^ParamBlockRec;

{20 bytes of system parameter area}

SysParmType =

PACKED RECORD

Valid: Byte; {validation field (\$A7)}

ATalkA: Byte; {AppleTalk node number hint for port A }

ATalkB: Byte; {AppleTalk node number hint for port B }
 Config: Byte; {ATalk port configuration A = bits 4-7, B = 0-3}
 PortA: Integer; {SCC port A configuration}
 PortB: Integer; {SCC port B configuration}
 Alarm: LongInt; {alarm time}
 Font: Integer; {default font id}
 KbdPrint: Integer; {high byte = kbd repeat}
 {high nibble = thresh in 4/60ths}
 {low nibble = rates in 2/60ths}
 {low byte = print stuff}
 VolClik: Integer; {low 3 bits of high byte = volume control}
 {high nibble of low byte = double time in 4/60ths}
 {low nibble of low byte = caret blink time in 4/60ths}
 Misc: Integer; {EEEE EEEE PSKB FFHH}
 {E = extra}
 {P = paranoia level}
 {S = mouse scaling}
 {K = key click}
 {B = boot disk}
 {F = menu flash}
 {H = help level}
 END; {SysParmType}

SysPPtr = ^SysParmType;

{volume control block data structure}

VCB =

RECORD

qLink: QElemPtr; {link to next element}
 qType: INTEGER; {not used}
 vcbFlags: INTEGER;
 vcbSigWord: INTEGER;
 vcbCrDate: LongInt;
 vcbLsMod: LongInt;
 vcbAtrb: INTEGER;
 vcbNmFls: INTEGER;


```

vcvVBMSt: INTEGER;
vcvAllocPtr: INTEGER;
vcvNmAIBlks: INTEGER;
vcvAIBlkSiz: LongInt;
vcvClpSiz: LongInt;
vcvAIBlSt: INTEGER;
vcvNxtCNID: LongInt;
vcvFreeBks: INTEGER;
vcvVN: STRING[27];
vcvDrvNum: INTEGER;
vcvDRefNum: INTEGER;
vcvFSID: INTEGER;
vcvVRefNum: INTEGER;
vcvMAdr: Ptr;
vcvBufAdr: Ptr;
vcvMLen: INTEGER;
vcvDirIndex: INTEGER;
vcvDirBlk: INTEGER;
    {new HFS extensions}
vcvVolBkup: LONGINT;
vcvVSegNum: INTEGER;
vcvWrCnt: LONGINT;
vcvXTClpSiz: LONGINT;
vcvCTClpSiz: LONGINT;
vcvNmRtDirs: INTEGER;
vcvFilCnt: LONGINT;
vcvDirCnt: LONGINT;
vcvFndrInfo: FInfo;
vcvVCSiz: INTEGER;
vcvVBMCSiz: INTEGER;
vcvCtlCSiz: INTEGER;
    {additional VCB info}
vcvXTAIBlks: INTEGER;
vcvCTAIBlks: INTEGER;
vcvXTRef: INTEGER;
vcvCTRef: INTEGER;

```



```
vcbCtlBuf: LONGINT;
vcbDirIDM: LONGINT;
vcbOffsM: INTEGER;
END;
```

{general queue data structure}

QHdr =

RECORD

```
QFlags: Integer;      {misc flags}
QHead: QElemPtr;      {first elem}
QTail: QElemPtr;      {last elem}
END; {QHdr}
```

QHdrPtr = ^QHdr;

{there are currently 4 types of queues:}

{VType - queue of Vertical Blanking Control Blocks}

{IOQType - queue of I/O queue elements}

{DrvType - queue of drivers}

{EvType - queue of Event Records}

{FSQType - queue of VCB elements}

{TimerType no longer is used. DrvType replaces it here in enum type}

QTypes = (dummyType,vType,ioQType,drvQType,evType,fsQType);

QElem =

RECORD

CASE QTypes OF

vType:

(vblQelem: VBLTask); {vertical blanking}

ioQType:

(ioQElem: ParamBlockRec); {I/O parameter block}

drvQType:

(drvQElem: DrvQEI); {drive}

evType:

(evQElem: EvQEI); {event}


```
fsQType:
    (vcbQElem: VCB); {volume control block}
END; {QElem}
```

{device control entry}

DCtlEntry =

RECORD

DCtlDriver: Ptr; {ptr to ROM or handle to RAM driver}

DCtlFlags: Integer; {flags}

DCtlQHdr: QHdr; {driver's i/o queue}

DCtlPosition: LongInt; {byte pos used by read and write calls}

DCtlStorage: Handle; {hndl to RAM drivers private storage}

DCtlRefNum: Integer; {driver's reference number}

DCtlCurTicks: LongInt; {long counter for timing system task calls}

DCtlWindow: Ptr; {ptr to driver's window if any}

DCtlDelay: Integer; {number of ticks btwn sysTask calls}

DCtlEMask: Integer; {desk acessory event mask}

DCtlMenu: Integer; {menu ID of menu associated with driver}

END;

DCtlPtr = ^DCtlEntry;

DCtlHandle = ^DCtlPtr;

{for Serial Driver}

SerShk =

PACKED RECORD {handshake control fields}

fXOn: Byte; {XON flow control enabled flag}

fCTS: Byte; {CTS flow control enabled flag}

xon: Char; {XOn character}

xoff: Char; {XOff character}

errs: Byte; {errors mask bits}

evts: Byte; {event enable mask bits}

fInX: Byte; {Input flow control enabled flag}

null: Byte; {unused}

END;

{parameter block structure for file and IO routines}

```
SerStaRec =
  PACKED RECORD
  cumErrs: Byte; {cumulative errors report}
  XOFFSent: Byte; {XOff Sent flag}
  rdPend: Byte; {read pending flag}
  wrPend: Byte; {write pending flag}
  ctsHold: Byte; {CTS flow control hold flag}
  XOFFHold: Byte; {XOff flow control hold flag}
  END;
```

{for Sound Driver}

{for 4-tone sound generation}

```
Wave = PACKED ARRAY[0..255] OF Byte;
```

```
WavePtr = ^Wave;
```

```
FTSoundRec =
```

```
  RECORD
```

```
  duration: Integer;
```

```
  sound1Rate: LongInt;
```

```
  sound1Phase: LongInt;
```

```
  sound2Rate: LongInt;
```

```
  sound2Phase: LongInt;
```

```
  sound3Rate: LongInt;
```

```
  sound3Phase: LongInt;
```

```
  sound4Rate: LongInt;
```

```
  sound4Phase: LongInt;
```

```
  sound1Wave: WavePtr;
```

```
  sound2Wave: WavePtr;
```

```
  sound3Wave: WavePtr;
```

```
  sound4Wave: WavePtr;
```

```
  END;
```

```
FTSndRecPtr = ^FTSoundRec;
```

```
FTSynthRec =
```

```
  RECORD
```



```

mode: Integer;
sndRec: FTSndRecPtr;
END;
FTSynthPtr = ^FTSynthRec;

Tone =
  RECORD
    count: Integer;
    amplitude: Integer;
    duration: Integer;
  END;

Tones = ARRAY[0..5000] OF Tone;

SWSynthRec =
  RECORD
    mode: Integer;
    triplets: Tones;
  END;
SWSynthPtr = ^SWSynthRec;

freeWave = PACKED ARRAY[0..30000] OF Byte;

FFSynthRec =
  RECORD
    mode: Integer;
    count: Fixed;
    waveBytes: freeWave;
  END;

FFSynthPtr = ^FFSynthRec;

{for date and time}
DateTimeRec =

```


RECORD

Year, {1904,1905,...}
 Month, {1,...,12 corresponding to Jan,...,Dec}
 Day, {1,...,31}
 Hour, {0,...,23}
 Minute, {0,...,59}
 Second, {0,...,59}
 DayOfWeek: Integer; {1,...,7 corresponding to Sun,...,Sat}
 END; {DateTimeRec}

{for application parameter}

appFile =

RECORD

vRefNum: Integer;
 ftype: OSType;
 versNum: Integer; {versNum in high byte}
 fName: str255;
 END; {appFile}

{for RAM serial driver}

SPortSel = (SPortA,SPortB);

{for disk driver}

DrvSts =

RECORD

track: Integer; {current track}
 writeProt: SignedByte; {bit 7=1 if volume is locked}
 diskInPlace: SignedByte; {disk in place}
 installed: SignedByte; {drive installed}
 sides: SignedByte; {bit 7=0 if single-sided drive}
 qLink: QElemPtr; {next queue entry}
 qType: Integer; {not used}
 dqDrive: Integer; {drive number}
 dqRefNum: Integer; {driver reference number}
 dqFSID: Integer; {file-system identifier}
 twoSideFmt: SignedByte; {-1 if two-sided disk}

needsFlush: SignedByte; {reserved}

diskErrs: Integer; {error count}

END; {DrvSts}

{ Event Manager Routines }

FUNCTION PostEvent(eventNum: Integer; eventMsg: LongInt): OsErr;
EXTERNAL;

{i}PROCEDURE FlushEvents(whichMask,stopMask: Integer);
EXTERNAL;

{i}PROCEDURE SetEventMask(theMask: Integer); EXTERNAL;

FUNCTION OSEventAvail(mask: Integer;

VAR theEvent: EventRecord): Boolean; EXTERNAL;

FUNCTION GetOSEvent(mask: Integer;

VAR theEvent: EventRecord): Boolean; EXTERNAL;

{ OS utilities Routines }

FUNCTION HandToHand(VAR theHndl: Handle): OsErr; EXTERNAL;

FUNCTION PtrToXHand(srcPtr: Ptr; dstHndl: Handle;

size: LongInt): OsErr; EXTERNAL;

FUNCTION PtrToHand(srcPtr: Ptr; VAR dstHndl: Handle;

size: LongInt): OsErr; EXTERNAL;

FUNCTION HandAndHand(hand1,hand2: Handle): OsErr;

EXTERNAL;

FUNCTION PtrAndHand(ptr1: Ptr; hand2: Handle;

size: LongInt): OsErr; EXTERNAL;

PROCEDURE SysBeep(duration: Integer); INLINE \$A9C8;

PROCEDURE Environs(VAR rom,machine: Integer); EXTERNAL;

PROCEDURE Restart; EXTERNAL;

{ Memory Manager Routines }

PROCEDURE SetApplBase(startPtr: Ptr); EXTERNAL;

PROCEDURE InitApplZone; EXTERNAL;

PROCEDURE InitZone(pgrowZone: ProcPtr; cmoreMasters: Integer;

limitPtr,startPtr : Ptr); EXTERNAL;

FUNCTION GetZone: THz; EXTERNAL;

PROCEDURE SetZone(hz: THz); EXTERNAL;


```

PROCEDURE HNoPurge(h: Handle); EXTERNAL;
{}FUNCTION ApplicZone: THz; EXTERNAL;
{}FUNCTION SystemZone: THz; EXTERNAL;
PROCEDURE CompactMem(cbNeeded: Size): Size; EXTERNAL;
PROCEDURE PurgeMem(cbNeeded: Size); EXTERNAL;
FUNCTION FreeMem: LongInt; EXTERNAL;
PROCEDURE ResrvMem(cbNeeded: Size); EXTERNAL;
FUNCTION MaxMem(Var grow: Size): Size; EXTERNAL;
{}FUNCTION TopMem: Ptr; EXTERNAL;

PROCEDURE SetGrowZone(growZone: ProcPtr); EXTERNAL;
PROCEDURE SetApplLimit(zoneLimit: Ptr); EXTERNAL;
{}FUNCTION GetApplLimit: Ptr; EXTERNAL;
PROCEDURE MaxApplZone; EXTERNAL;
PROCEDURE MoveHHi (h: handle); EXTERNAL;

FUNCTION NewPtr(byteCount: Size): Ptr; EXTERNAL;
PROCEDURE DisposPtr(p: Ptr); EXTERNAL;
FUNCTION GetPtrSize(p: Ptr): Size; EXTERNAL;
PROCEDURE SetPtrSize(p: Ptr; newSize: Size); EXTERNAL;
FUNCTION PtrZone(p: Ptr): THz; EXTERNAL;

FUNCTION NewHandle(byteCount: Size): Handle; EXTERNAL;
PROCEDURE DisposHandle(h: Handle); EXTERNAL;
FUNCTION GetHandleSize(h: Handle): Size; EXTERNAL;
PROCEDURE SetHandleSize(h: Handle; newSize: Size); EXTERNAL;
FUNCTION HandleZone(h: Handle): THz; EXTERNAL;
FUNCTION RecoverHandle(p: Ptr): Handle; EXTERNAL;
PROCEDURE EmptyHandle(h: Handle); EXTERNAL;
PROCEDURE ReAllocHandle(h: Handle; byteCount: Size);
EXTERNAL;

PROCEDURE HLock(h: Handle); EXTERNAL;
PROCEDURE HUnLock(h: Handle); EXTERNAL;
PROCEDURE HPurge(h: Handle); EXTERNAL;

```



```

PROCEDURE HNoPurge(h: Handle);          EXTERNAL;
PROCEDURE MoreMasters;                   EXTERNAL;

PROCEDURE BlockMove(srcPtr, destPtr: Ptr; byteCount: Size);
EXTERNAL;
{;}FUNCTION MemError: OsErr;              EXTERNAL;

{;}FUNCTION GZSaveHnd: Handle;            EXTERNAL;

{ 128K ROM Memory Manager Routines }

FUNCTION StackSpace: LongInt;             EXTERNAL;
PROCEDURE PurgeSpace(VAR purgeFree, afterPurge: LongInt);
EXTERNAL;
FUNCTION MaxBlock: LongInt;               EXTERNAL;
FUNCTION NewEmptyHandle: Handle;          EXTERNAL;
FUNCTION GetMaskTable: Ptr;               EXTERNAL;
PROCEDURE HSetRBit(h: Handle);            EXTERNAL;
PROCEDURE HClrRBit(h: Handle);            EXTERNAL;
PROCEDURE HSetState(h: Handle; state: SignedByte); EXTERNAL;
FUNCTION HGetState(h: Handle): SignedByte; EXTERNAL;

{ Vertical Retrace Manager Routines }

FUNCTION VInstall(VBLTaskPtr: QElemPtr): OsErr; EXTERNAL;
FUNCTION VRemove(VBLTaskPtr: QElemPtr): OsErr; EXTERNAL;

{ Operating System Dispatcher Routines }

FUNCTION GetTrapAddress(trapNum: Integer): LongInt;
EXTERNAL;
PROCEDURE SetTrapAddress(trapAddr: LongInt; trapNum: Integer);
EXTERNAL;

{ Utility Core Routines }

```



```

FUNCTION GetSysPPtr: SysPPtr;          EXTERNAL;
FUNCTION WriteParam: OsErr;            EXTERNAL;
FUNCTION SetDateTime(time: LongInt):OsErr;    EXTERNAL;
FUNCTION ReadDateTime(VAR time: LongInt):OsErr;    EXTERNAL;
PROCEDURE GetDateTime(VAR secs: LongInt);    EXTERNAL;
PROCEDURE SetTime(d: DateTimeRec);    EXTERNAL;
PROCEDURE GetTime(VAR d: DateTimeRec);    EXTERNAL;
PROCEDURE Date2Secs(d: DateTimeRec; VAR s: LongInt);
EXTERNAL;
PROCEDURE Secs2Date(s: LongInt; VAR d: DateTimeRec);
EXTERNAL;
PROCEDURE Delay(numTicks: LongInt; VAR finalTicks: LongInt);
EXTERNAL;
FUNCTION EqualString(str1,str2: Str255;
    caseSens,diacSens: Boolean):Boolean;    EXTERNAL;
PROCEDURE UpString(VAR theString: Str255; diacSens: Boolean);
EXTERNAL;
FUNCTION InitUtil: OsErr;            EXTERNAL;

```

```
{ 128K ROM Utility Core Routines }
```

```

FUNCTION RelString(str1,str2: Str255;
    caseSens, diacSens: Boolean): INTEGER;    EXTERNAL;

```

```
{ Segment Loader Routines }
```

```

PROCEDURE UnLoadSeg(routineAddr: Ptr);    INLINE $A9F1;
PROCEDURE ExitToShell;    INLINE $A9F4;
PROCEDURE GetAppParms(VAR apName: str255; VAR apRefNum:
Integer;
    VAR apParam: Handle);    INLINE $A9F5;
PROCEDURE CountAppFiles(VAR message: Integer;
    VAR count: Integer);    EXTERNAL;
PROCEDURE GetAppFiles(index: Integer; VAR theFile: AppFile);
EXTERNAL;
PROCEDURE ClrAppFiles(index: Integer);    EXTERNAL;

```


{ Queue Routines }

```

PROCEDURE FInitQueue;           EXTERNAL;
PROCEDURE Enqueue(qElement: QElemPtr; qHeader: QHdrPtr);
EXTERNAL;
FUNCTION Dequeue(qElement: QElemPtr; qHeader: QHdrPtr): OsErr;
EXTERNAL;
FUNCTION GetFSQHdr: QHdrPtr;     EXTERNAL;
FUNCTION GetDrvQHdr: QHdrPtr;    EXTERNAL;
FUNCTION GetVCBQHdr: QHdrPtr;    EXTERNAL;
FUNCTION GetVBLQHdr: QHdrPtr;    EXTERNAL;
FUNCTION GetEvQHdr: QHdrPtr;     EXTERNAL;
FUNCTION GetDCtlEntry(refNum: Integer): DCtlHandle;
EXTERNAL;

```

{ File Manager Routines }

```

FUNCTION PBOpen(paramBlock: ParmBlkPtr; aSync: Boolean): OsErr;
EXTERNAL;
FUNCTION PBClose(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBRead(paramBlock: ParmBlkPtr; aSync: Boolean): OsErr;
EXTERNAL;
FUNCTION PBWrite(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBControl(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBStatus(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBKillIO(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBGetVInfo(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBGetVol(paramBlock: ParmBlkPtr; aSync: Boolean):

```



```

OsErr; EXTERNAL;
FUNCTION PBSetVol(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBFlushVol(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBCreate(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBDelete(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBOpenRF(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBRename(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBGetFInfo(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBSetFInfo(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBSetFLock(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBRstFLock(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBSetFVers(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBAllocate(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBGetEOF(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBSetEOF(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBGetFPos(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBSetFPos(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBFlushFile(paramBlock: ParmBlkPtr; aSync: Boolean):
OsErr; EXTERNAL;
FUNCTION PBMountVol(paramBlock: ParmBlkPtr): OsErr;

```



```

EXTERNAL;
FUNCTION PBUntMountVol(paramBlock: ParmBlkPtr): OsErr;
EXTERNAL;
FUNCTION PBEject(paramBlock: ParmBlkPtr): OsErr; EXTERNAL;
FUNCTION PBOffLine(paramBlock: ParmBlkPtr): OsErr;
EXTERNAL;
PROCEDURE AddDrive(drvrRefNum: Integer; drvNum: Integer;
    QEI: drvQEIPtr); EXTERNAL;

```

{ File Manager High Level FS Calls }

```

FUNCTION FSOpen(fileName: Str255; vRefNum: Integer;
    VAR refNum: Integer): OsErr; EXTERNAL;
FUNCTION FSClose(refNum: Integer): OsErr; EXTERNAL;
FUNCTION FSRead(refNum: Integer; VAR count: LongInt;
    buffPtr: Ptr): OsErr; EXTERNAL;
FUNCTION FSWrite(refNum: Integer; VAR count: LongInt;
    buffPtr: Ptr): OsErr; EXTERNAL;
FUNCTION Control(refNum: Integer; csCode: Integer;
    csParamPtr: Ptr): OsErr; EXTERNAL;
FUNCTION Status(refNum: Integer; csCode: Integer;
    csParamPtr: Ptr): OsErr; EXTERNAL;
FUNCTION KillIO(refNum: Integer): OsErr; EXTERNAL;

```

{ File Manager Volume Calls }

```

FUNCTION GetVInfo(drvNum: Integer; volName: StringPtr; VAR
    vRefNum: Integer;
    VAR FreeBytes: LongInt): OsErr; EXTERNAL;
FUNCTION GetFInfo(fileName: Str255; vRefNum: Integer;
    VAR FndrInfo: FInfo): OsErr; EXTERNAL;
FUNCTION GetVol(volName: StringPtr;
    VAR vRefNum: Integer): OsErr; EXTERNAL;
FUNCTION SetVol(volName: StringPtr; vRefNum: Integer): OsErr;
EXTERNAL;
FUNCTION UnMountVol(volName: StringPtr;

```



```

    vRefNum: Integer):OsErr;    EXTERNAL;
FUNCTION  Eject(volName: StringPtr; vRefNum: Integer): OsErr;
EXTERNAL;
FUNCTION  FlushVol(volName: StringPtr; vRefNum: Integer):OsErr;
EXTERNAL;
{ File Manager Unopened file calls }

FUNCTION  Create(fileName: Str255; vRefNum: Integer; creator:
OSType;
    fileType: OSType):OsErr;    EXTERNAL;
FUNCTION  FSDelete(fileName: Str255; vRefNum: Integer):OsErr;
EXTERNAL;
FUNCTION  OpenRF(fileName: Str255; vRefNum: Integer;
    VAR refNum: Integer): OsErr;    EXTERNAL;
FUNCTION  Rename(oldName: Str255; vRefNum: Integer;
    newName: Str255):OsErr;    EXTERNAL;
FUNCTION  SetFInfo(fileName: Str255; vRefNum: Integer;
    FndrInfo: FInfo):OsErr;    EXTERNAL;
FUNCTION  SetFLock(fileName: Str255; vRefNum: Integer):OsErr;
EXTERNAL;
FUNCTION  RstFLock(fileName: Str255; vRefNum: Integer):OsErr;
EXTERNAL;
{ File Manager, Opened file calls }

FUNCTION  Allocate(refNum: Integer; VAR count: LongInt):OsErr;
EXTERNAL;
FUNCTION  GetEOF(refNum: Integer; VAR LogEOF: LongInt):OsErr;
EXTERNAL;
FUNCTION  SetEOF(refNum: Integer; LogEOF: LongInt):OsErr;
EXTERNAL;
FUNCTION  GetFPos(refNum: Integer; VAR filePos: LongInt):OsErr;
EXTERNAL;
FUNCTION  SetFPos(refNum: Integer; posMode: Integer;
    posOff: LongInt):OsErr;    EXTERNAL;

```



```
FUNCTION GetVRefNum(fileRefNum: Integer):Integer;    EXTERNAL;
```

```
PROCEDURE SysErrProc(errno: Integer);    EXTERNAL;
```

```
{ Serial Driver Interface }
```

```
FUNCTION OpenDriver(name: Str255;
```

```
    VAR drvRefNum: Integer): OsErr;    EXTERNAL;
```

```
FUNCTION CloseDriver(refNum: Integer):OsErr;    EXTERNAL;
```

```
FUNCTION SerReset(refNum: Integer; serConfig: Integer): OsErr;  
EXTERNAL;
```

```
FUNCTION SerSetBuf(refNum: Integer; serBPtr: Ptr;  
    serBLen: Integer): OsErr;    EXTERNAL;
```

```
FUNCTION SerHShake(refNum: Integer; flags: SerShk): OsErr;  
EXTERNAL;
```

```
FUNCTION SerSetBrk(refNum: Integer): OsErr;    EXTERNAL;
```

```
FUNCTION SerClrBrk(refNum: Integer): OsErr;    EXTERNAL;
```

```
FUNCTION SerGetBuf(refNum: Integer;  
    VAR count: LongInt): OsErr;    EXTERNAL;
```

```
FUNCTION SerStatus(refNum: Integer;  
    VAR serSta: SerStaRec): OsErr;    EXTERNAL;
```

```
FUNCTION DiskEject(drvnum: Integer): OsErr;    EXTERNAL;
```

```
FUNCTION SetTagBuffer(buffPtr: Ptr): OsErr;    EXTERNAL;
```

```
FUNCTION DriveStatus(drvNum: Integer;  
    VAR status: DrvSts): OsErr;    EXTERNAL;
```

```
FUNCTION RamSDOpen(whichPort: SPortSel): OsErr; EXTERNAL;
```

```
PROCEDURE RamSDClose(whichPort: SPortSel);    EXTERNAL;
```

```
{ Sound Driver }
```

```
PROCEDURE SetSoundVol(level: Integer);    EXTERNAL;
```

```
PROCEDURE GetSoundVol(VAR level: Integer);    EXTERNAL;
```

```
PROCEDURE StartSound(synthRec: Ptr; numBytes: LongInt;  
    CompletionRtn: ProcPtr);    EXTERNAL;
```

```
PROCEDURE StopSound;    EXTERNAL;
```

```
FUNCTION SoundDone: Boolean;    EXTERNAL;
```

```
{ System error handler }
```


PROCEDURE SysError(errorCode: Integer); EXTERNAL;

NAME

QuickDraw -- QuickDraw Interface (18 Jan 85)

DESCRIPTION

This file contains the interface to the QuickDraw routines.
All of the interface routines are implemented as in-line traps.

For more detailed information see the "QuickDraw" section of
the "Inside Macintosh" documentation.

AUTHOR

Bill Atkinson

Copyright 1983, 1984, 1985 Apple Computer Inc.

MODIFICATIONS

Tom Leonard, TML Systems

Changed to "include file" format for use with MacLanguage

Series: Pascal

Additions for the 128K ROMs are added February 1985.

DO NOT attempt to use these with the 64K ROMs.

CONST srcCopy = 0; (the 16 transfer modes)

srcOr = 1;

srcXor = 2;

srcBis = 3;

notSrcCopy = 4;

notSrcOr = 5;

notSrcXor = 6;

notSrcBis = 7;



{-----}

QUICKDRAW Pascal Interface QUICKDRAW

NAME

QuickDraw -- QuickDraw Interface (16 Jan 85)

DESCRIPTION

This file contains the interface to the QuickDraw routines.
All of the interface routines are implemented as in-line traps.

For more detailed information see the "QuickDraw" section of
the "Inside Macintosh" documentation.

AUTHOR

Bill Atkinson
Copyright 1983, 1984, 1985 Apple Computer Inc.

MODIFICATIONS

Tom Leonard, TML Systems

Changed to "include file" format for use with MacLanguage
Series Pascal

Additions for the 128K ROMs are added February 1986.

DO NOT attempt to use these with the 64K ROMs.

-----}

CONST srcCopy = 0; { the 16 transfer modes }

srcOr = 1;

srcXor = 2;

srcBic = 3;

notSrcCopy = 4;

notSrcOr = 5;

notSrcXor = 6;

notSrcBic = 7;

Pr 4 - 2

```
patCopy    = 8;  
patOr      = 9;  
patXor     = 10;  
patBic     = 11;  
notPatCopy = 12;  
notPatOr   = 13;  
notPatXor  = 14;  
notPatBic  = 15;
```

{ QuickDraw color separation constants }

```
normalBit   = 0;      { normal screen mapping }  
inverseBit  = 1;      { inverse screen mapping }  
redBit      = 4;      { RGB additive mapping }  
greenBit    = 3;  
blueBit     = 2;  
cyanBit     = 8;      { CMYBk subtractive mapping }  
magentaBit  = 7;  
yellowBit   = 6;  
blackBit    = 5;
```

```
blackColor  = 33;      { colors expressed in these mappings }  
whiteColor  = 30;  
redColor    = 205;  
greenColor  = 341;  
blueColor   = 409;  
cyanColor   = 273;  
magentaColor = 137;  
yellowColor = 69;
```

```
picLParen   = 0;      { standard picture comments }  
picRParen   = 1;
```

```
TYPE QDByte = SignedByte;  
QDPtr      = Ptr;    { blind pointer }
```


Pr 4 - 3

QDHandle = Handle; { blind handle }

Pattern = PACKED ARRAY[0..7] OF 0..255;

Bits16 = ARRAY[0..15] OF INTEGER;

VHSelect = (v,h);

GrafVerb = (frame,paint,erase,invert,fill);

StyleItem = (bold,italic,underline,outline,shadow,condense,extend);

Style = SET OF StyleItem;

FontInfo = RECORD

ascent: INTEGER;

descent: INTEGER;

widMax: INTEGER;

leading: INTEGER;

END;

Point = RECORD CASE INTEGER OF

0: (v: INTEGER;

h: INTEGER);

1: (vh: ARRAY[VHSelect] OF INTEGER);

END;

Rect = RECORD CASE INTEGER OF

0: (top: INTEGER; OF Point;

left: INTEGER;

bottom: INTEGER;

right: INTEGER);

1: (topLeft: Point;

botRight: Point);

END;

BitMap = RECORD { not rectangular }

baseAddr: Ptr;

rowBytes: INTEGER;


```

    bounds: Rect;
END;

```

```

Cursor = RECORD

```

```

    data: Bits16;

```

```

    mask: Bits16;

```

```

    hotSpot: Point;

```

```

END;

```

```

PenState = RECORD

```

```

    pnLoc: Point;

```

```

    pnSize: Point;

```

```

    pnMode: INTEGER;

```

```

    pnPat: Pattern;

```

```

END;

```

```

PolyHandle = ^PolyPtr;

```

```

PolyPtr = ^Polygon;

```

```

Polygon = RECORD

```

```

    polySize: INTEGER;

```

```

    polyBBox: Rect;

```

```

    polyPoints: ARRAY[0..0] OF Point;

```

```

END;

```

```

RgnHandle = ^RgnPtr;

```

```

RgnPtr = ^Region;

```

```

Region = RECORD

```

```

    rgnSize: INTEGER; { rgnSize = 10 for rectangular }

```

```

    rgnBBox: Rect;

```

```

    { plus more data if not rectangular }

```

```

END;

```



```

PicHandle = ^PicPtr;
PicPtr    = ^Picture;
Picture = RECORD
    picSize:  INTEGER;
    picFrame: Rect;
    { plus byte codes for picture content }
END;

```

```

QDProcsPtr = ^QDProcs;
QDProcs = RECORD
    textProc:  Ptr;
    lineProc:  Ptr;
    rectProc:  Ptr;
    rRectProc: Ptr;
    ovalProc:  Ptr;
    arcProc:   Ptr;
    polyProc:  Ptr;
    rgnProc:   Ptr;
    bitsProc:  Ptr;
    commentProc: Ptr;
    txMeasProc: Ptr;
    getPicProc: Ptr;
    putPicProc: Ptr;
    END;

```

```

GrafPtr = ^GrafPort;
GrafPort = RECORD
    device:  INTEGER;
    portBits: BitMap;
    portRect: Rect;
    visRgn:  RgnHandle;
    clipRgn: RgnHandle;
    bkPat:   Pattern;

```



```

fillPat:   Pattern;
pnLoc:     Point;
pnSize:    Point;
pnMode:    INTEGER;
pnPat:     Pattern;
pnVis:     INTEGER;
txFont:    INTEGER;
txFace:    Style;
txMode:    INTEGER;
txSize:    INTEGER;
spExtra:   LongInt;
fgColor:   LongInt;
bkColor:   LongInt;
colrBit:   INTEGER;
patStretch: INTEGER;
picSave:   Handle;
rgnSave:   Handle;
polySave:  Handle;
grafProcs: QDProcsPtr;
END;

```

```
TYPE FMOOutPtr = ^FMOOutRec;
```

```
FMOOutrec = PACKED RECORD
```

```

    errNum:   INTEGER;    { used only for GrafError  }
    fontHandle: QDHandle;  { handle to font      }
    bold:     Byte;       { how much to smear horiz  }
    italic:   Byte;       { how much to shear      }
    ulOffset: Byte;       { pixels below baseline  }
    ulShadow: Byte;       { how big is the halo    }
    ulThick:  Byte;       { how thick is the underline }
    shadow:   Byte;       { 0,1,2,or 3 only      }
    extra:    SignedByte; { extra white dots each char }
    ascent:   Byte;       { ascent measure for font  }
    descent:  Byte;       { descent measure for font }
    widMax:   Byte;       { width of widest char    }
    leading:  SignedByte; { leading between lines   }

```


Pr 4 - 7

```

gray:    unused:  Byte;
black:   number:  Point;    { use this modified scale to }
white:   denom:   Point;    { draw or measure text with }
theFont: END;

```

{ The following are the QuickDraw variables. They are declared here in reverse

order of their normal appearance. This is because the MDS linker allocates

storage for globals (relative to A5) in reverse order of their declartion.

```

}
PROCEDURE ClosePort (port: GrafPtr);
PROCEDURE SetPort (port: GrafPtr);
VAR { Private QuickDraw variables }

```

```

fontData: FMOOutRec;
fontPtr:  FMOOutPtr;    { the last font used, used by DrawText }
fontAdj:  LongInt;      { Fixed Point to adjust for font subst. }
patAlign: Point;        { to align pattern during DrawPicture }
polyMax:  INTEGER;      { max bytes allocated so far to thePoly }
thePoly:  PolyHandle;   { the current polygon being defined }
playIndex: INTEGER;     { used by StdGetPic }
playPic:  PicHandle;    { used by StdGetPic }
rgnMax:   INTEGER;      { max bytes allocated so far to rgnBuf }
rgnIndex: INTEGER;      { current bytes used in rgnBuf }
rgnBuf:   QDHandle;     { point saving buffer for OpenRgn }
wideData: Region;
wideMaster: RgnPtr;
wideOpen:  RgnHandle;   { a dummy rectangular region,
read-only }

```

```

{ Public QuickDraw variables }
randSeed: LongInt;
screenBits: BitMap;
arrow:     Cursor;
dkGray:    Pattern;
ltGray:    Pattern;

```



```

gray:   Pattern;
black:  Pattern;
white:  Pattern;
thePort: GrafPtr;

```

{ GrafPort Routines }

```

PROCEDURE InitGraf (globalPtr: Ptr);      INLINE $A86E;
PROCEDURE OpenPort (port: GrafPtr);      INLINE $A86F;
PROCEDURE InitPort (port: GrafPtr);      INLINE $A86D;
PROCEDURE ClosePort (port: GrafPtr);     INLINE $A87D;
PROCEDURE SetPort (port: GrafPtr);       INLINE $A873;
PROCEDURE GetPort (VAR port: GrafPtr);    INLINE $A874;
PROCEDURE GrafDevice (device: INTEGER);   INLINE $A872;
PROCEDURE SetPortBits(bm: BitMap);        INLINE $A875;
PROCEDURE PortSize (width,height: INTEGER);  INLINE $A876;
PROCEDURE MovePortTo (leftGlobal,topGlobal: INTEGER);  INLINE
$A877;
PROCEDURE SetOrigin (h,v: INTEGER);       INLINE $A878;
PROCEDURE SetClip (rgn: RgnHandle);       INLINE $A879;
PROCEDURE GetClip (rgn: RgnHandle);       INLINE $A87A;
PROCEDURE ClipRect (r: Rect);             INLINE $A87B;
PROCEDURE BackPat (pat: Pattern);         INLINE $A87C;

```

{ Cursor Routines }

```

PROCEDURE InitCursor;                     INLINE $A850;
PROCEDURE SetCursor(crsr: Cursor);        INLINE $A851;
PROCEDURE HideCursor;                     INLINE $A852;
PROCEDURE ShowCursor;                     INLINE $A853;
PROCEDURE ObscureCursor;                   INLINE $A856;

```

{ Line Routines }

```

PROCEDURE HidePen;                        INLINE $A896;

```


Pr 4 - 9

```
PROCEDURE ShowPen;          INLINE $A897;
PROCEDURE GetPen  (VAR pt: Point);      INLINE $A89A;
PROCEDURE GetPenState(VAR pnState: PenState);      INLINE
$A898;
PROCEDURE SetPenState(pnState: PenState);      INLINE $A899;
PROCEDURE PenSize  (width,height: INTEGER);      INLINE $A89B;
PROCEDURE PenMode  (mode: INTEGER);      INLINE $A89C;
PROCEDURE PenPat   (pat: Pattern);      INLINE $A89D;
PROCEDURE PenNormal;      INLINE $A89E;
PROCEDURE MoveTo   (h,v: INTEGER);      INLINE $A893;
PROCEDURE Move     (dh,dv: INTEGER);      INLINE $A894;
PROCEDURE LineTo   (h,v: INTEGER);      INLINE $A891;
PROCEDURE Line     (dh,dv: INTEGER);      INLINE $A892;
```

{ Text Routines }

```
PROCEDURE TextFont  (font: INTEGER);      INLINE $A887;
PROCEDURE TextFace  (face: Style);      INLINE $A888;
PROCEDURE TextMode  (mode: INTEGER);      INLINE $A889;
PROCEDURE TextSize  (size: INTEGER);      INLINE $A88A;
PROCEDURE SpaceExtra (extra: Fixed);      INLINE $A88E;
PROCEDURE DrawChar  (ch: char);      INLINE $A883;
PROCEDURE DrawString (s: Str255);      INLINE $A884;
PROCEDURE DrawText   (textBuf: Ptr;
    firstByte,byteCount: INTEGER);      INLINE $A885;
FUNCTION CharWidth  (ch: CHAR): INTEGER;      INLINE $A88D;
FUNCTION StringWidth (s: Str255): INTEGER;      INLINE $A88C;
FUNCTION TextWidth   (textBuf: Ptr; firstByte,
    byteCount: INTEGER): INTEGER;      INLINE $A886;
PROCEDURE GetFontInfo (VAR info: FontInfo);      INLINE $A88B;
```

{ Point Calculations }

```
PROCEDURE AddPt      (src: Point; VAR dst: Point);      INLINE
$A87E;
```



```

PROCEDURE SubPt      (src: Point; VAR dst: Point);   INLINE
$A87F;
PROCEDURE SetPt      (VAR pt: Point; h,v: INTEGER);  INLINE
$A880;
FUNCTION EqualPt      (pt1,pt2: Point): BOOLEAN;    INLINE $A881;
PROCEDURE ScalePt     (VAR pt: Point;
  fromRect,toRect: Rect);  INLINE $A8F8;
PROCEDURE MapPt       (VAR pt: Point;
  fromRect,toRect: Rect);  INLINE $A8F9;
PROCEDURE LocalToGlobal (VAR pt: Point);            INLINE $A870;
PROCEDURE GlobalToLocal (VAR pt: Point);            INLINE $A871;

```

```

{ Rectangle Calculations }

```

```

PROCEDURE SetRect     (VAR r: Rect;
  left,top,right,bottom: INTEGER);  INLINE $A8A7;
FUNCTION EqualRect     (rect1,rect2: Rect): BOOLEAN;    INLINE
$A8A6;
FUNCTION EmptyRect     (r: Rect): BOOLEAN;              INLINE $A8AE;
PROCEDURE OffsetRect   (VAR r: Rect; dh,dv: INTEGER);   INLINE
$A8A8;
PROCEDURE MapRect      (VAR r: Rect;
  fromRect,toRect: Rect);  INLINE $A8FA;
PROCEDURE InsetRect    (VAR r: Rect; dh,dv: INTEGER);   INLINE
$A8A9;
FUNCTION SectRect      (src1,src2: Rect;
  VAR dstRect: Rect): BOOLEAN;  INLINE $A8AA;
PROCEDURE UnionRect    (src1,src2: Rect;
  VAR dstRect: Rect);  INLINE $A8AB;
FUNCTION PtInRect      (pt: Point; r: Rect): BOOLEAN;   INLINE
$A8AD;
PROCEDURE Pt2Rect      (pt1,pt2: Point;
  VAR dstRect: Rect);  INLINE $A8AC;

```

```

{ Graphical Operations on Rectangles }

```



```

PROCEDURE FrameRect (r: Rect);          INLINE $A8A1;
PROCEDURE PaintRect (r: Rect);          INLINE $A8A2;
PROCEDURE EraseRect (r: Rect);          INLINE $A8A3;
PROCEDURE InvertRect (r: Rect);         INLINE $A8A4;
PROCEDURE FillRect (r: Rect; pat: Pattern);  INLINE $A8A5;

```

{ RoundRect Routines }

```

PROCEDURE FrameRoundRect (r: Rect; ovWd,ovHt: INTEGER);
  INLINE $A8B0;
PROCEDURE PaintRoundRect (r: Rect; ovWd,ovHt: INTEGER);
  INLINE $A8B1;
PROCEDURE EraseRoundRect (r: Rect; ovWd,ovHt: INTEGER);
  INLINE $A8B2;
PROCEDURE InvertRoundRect (r: Rect; ovWd,ovHt: INTEGER);
  INLINE $A8B3;
PROCEDURE FillRoundRect (r: Rect; ovWd,ovHt: INTEGER;
  pat: Pattern);  INLINE $A8B4;

```

{ Oval Routines }

```

PROCEDURE FrameOval (r: Rect);          INLINE $A8B7;
PROCEDURE PaintOval (r: Rect);          INLINE $A8B8;
PROCEDURE EraseOval (r: Rect);          INLINE $A8B9;
PROCEDURE InvertOval (r: Rect);         INLINE $A8BA;
PROCEDURE FillOval (r: Rect; pat: Pattern);  INLINE $A8BB;

```

{ Arc Routines }

```

PROCEDURE FrameArc (r: Rect; startAngle,arcAngle: INTEGER);
  INLINE $A8BE;
PROCEDURE PaintArc (r: Rect; startAngle,arcAngle: INTEGER);
  INLINE $A8BF;
PROCEDURE EraseArc (r: Rect; startAngle,arcAngle: INTEGER);
  INLINE $A8C0;

```



```

PROCEDURE InvertArc (r: Rect; startAngle,arcAngle: INTEGER);
  INLINE $A8C1;
PROCEDURE FillArc (r: Rect; startAngle,arcAngle: INTEGER;
  pat: Pattern);    INLINE $A8C2;
PROCEDURE PtToAngle (r: Rect; pt: Point; VAR angle: INTEGER);
  INLINE $A8C3;
PROCEDURE SetClipRgn (srcRgnA,srcRgnB,dstRgn: RgnHandle);
{ Polygon Routines }
PROCEDURE UnionRgn (srcRgnA,srcRgnB,dstRgn: RgnHandle);
FUNCTION OpenPoly: PolyHandle;    INLINE $A8CB;
PROCEDURE ClosePoly;    INLINE $A8CC;
PROCEDURE KillPoly (poly: PolyHandle);    INLINE $A8CD;
PROCEDURE OffsetPoly (poly: PolyHandle; dh,dv: INTEGER);
  INLINE $A8CE;
PROCEDURE MapPoly (poly: PolyHandle; fromRect,toRect:
  Rect);INLINE $A8FC;
PROCEDURE FramePoly (poly: PolyHandle);    INLINE $A8C6;
PROCEDURE PaintPoly (poly: PolyHandle);    INLINE $A8C7;
PROCEDURE ErasePoly (poly: PolyHandle);    INLINE $A8C8;
PROCEDURE InvertPoly (poly: PolyHandle);    INLINE $A8C9;
PROCEDURE FillPoly (poly: PolyHandle; pat: Pattern);    INLINE
  $A8CA;
{ Region Calculations }
PROCEDURE NewRgn: RgnHandle;    INLINE $A8D8;
PROCEDURE DisposeRgn(rgn: RgnHandle);    INLINE $A8D9;
PROCEDURE CopyRgn (srcRgn,dstRgn: RgnHandle);    INLINE
  $A8DC;
PROCEDURE SetEmptyRgn(rgn: RgnHandle);    INLINE $A8DD;
PROCEDURE SetRectRgn(rgn: RgnHandle;
  left,top,right,bottom: INTEGER);    INLINE $A8DE;
PROCEDURE RectRgn (rgn: RgnHandle; r: Rect);    INLINE
  $A8DF;
PROCEDURE OpenRgn;    INLINE $A8DA;
PROCEDURE CloseRgn (dstRgn: RgnHandle);    INLINE $A8DB;

```



```

PROCEDURE OffsetRgn (rgn: RgnHandle; dh,dv: INTEGER);
  INLINE $A8E0;
PROCEDURE MapRgn    (rgn: RgnHandle;  fromRect,toRect: Rect);
  INLINE $A8FB;
PROCEDURE InsetRgn (rgn: RgnHandle; dh,dv: INTEGER);      INLINE
  $A8E1;
PROCEDURE SectRgn  (srcRgnA,srcRgnB,dstRgn: RgnHandle);
  INLINE $A8E4;
PROCEDURE UnionRgn (srcRgnA,srcRgnB,dstRgn: RgnHandle);
  INLINE $A8E5;
PROCEDURE DiffRgn  (srcRgnA,srcRgnB,dstRgn: RgnHandle);
  INLINE $A8E6;
PROCEDURE XorRgn   (srcRgnA,srcRgnB,dstRgn: RgnHandle);
  INLINE $A8E7;
FUNCTION EqualRgn (rgnA,rgnB: RgnHandle): BOOLEAN;      INLINE
  $A8E3;
FUNCTION EmptyRgn (rgn: RgnHandle): BOOLEAN;           INLINE
  $A8E2;
FUNCTION PtInRgn  (pt: Point; rgn: RgnHandle): BOOLEAN;
  INLINE $A8E8;
FUNCTION RectInRgn (r: Rect; rgn: RgnHandle): BOOLEAN;
  INLINE $A8E9;

```

{ Graphical Operations on Regions }

```

PROCEDURE FrameRgn (rgn: RgnHandle);      INLINE $A8D2;
PROCEDURE PaintRgn (rgn: RgnHandle);      INLINE $A8D3;
PROCEDURE EraseRgn (rgn: RgnHandle);      INLINE $A8D4;
PROCEDURE InvertRgn (rgn: RgnHandle);     INLINE $A8D5;
PROCEDURE FillRgn  (rgn: RgnHandle; pat: Pattern);      INLINE
  $A8D6;

```

{ Graphical Operations on BitMaps }

```

PROCEDURE ScrollRect(dstRect: Rect; dh,dv: INTEGER;
  updateRgn: rgnHandle);  INLINE $A8EF;

```



```

PROCEDURE CopyBits (srcBits,dstBits: BitMap;
    srcRect,dstRect: Rect;
    mode: INTEGER;
    maskRgn: RgnHandle);    INLINE $A8EC;

```

```

{ Picture Routines }

```

```

FUNCTION OpenPicture(picFrame: Rect): PicHandle;    INLINE
$A8F3;
PROCEDURE ClosePicture;    INLINE $A8F4;
PROCEDURE DrawPicture(myPicture: PicHandle; dstRect: Rect);
    INLINE $A8F6;
PROCEDURE PicComment (kind,dataSize: INTEGER;
    dataHandle: Handle);    INLINE $A8F2;
PROCEDURE KillPicture(myPicture: PicHandle);    INLINE
$A8F5;

```

```

{ The Bottleneck Interface: }

```

```

PROCEDURE SetStdProcs(VAR procs: QDProcs);    INLINE
$A8EA;
PROCEDURE StdText (count: INTEGER; textAddr: Ptr;
    numer,denom: Point);    INLINE $A882;
PROCEDURE StdLine (newPt: Point);    INLINE $A890;
PROCEDURE StdRect (verb: GrafVerb; r: Rect);    INLINE
$A8A0;
PROCEDURE StdRRect (verb: GrafVerb; r: Rect;
    ovWd,ovHt: INTEGER);    INLINE $A8AF;
PROCEDURE StdOval (verb: GrafVerb; r: Rect);    INLINE
$A8B6;
PROCEDURE StdArc (verb: GrafVerb; r: Rect;
    startAngle,arcAngle: INTEGER);    INLINE $A8BD;
PROCEDURE StdPoly (verb: GrafVerb; poly: PolyHandle);
    INLINE $A8C5;
PROCEDURE StdRgn (verb: GrafVerb; rgn: RgnHandle);    INLINE
$A8D1;

```



```

PROCEDURE StdBits (VAR srcBits: BitMap;
    VAR srcRect,dstRect: Rect;
    mode: INTEGER; maskRgn: RgnHandle);    INLINE $A8EB;

```

```

PROCEDURE StdComment (kind,dataSize: INTEGER;
    dataHandle: Handle);    INLINE $A8F1;

```

```

FUNCTION StdTxMeas (count: INTEGER; textAddr: Ptr;
    VAR numer,denom: Point;

```

```

    VAR info: FontInfo): INTEGER;    INLINE $A8ED;
PROCEDURE StdGetPic (dataPtr: Ptr; byteCount: INTEGER);
    INLINE $A8EE;

```

```

PROCEDURE StdPutPic (dataPtr: Ptr; byteCount: INTEGER);
    INLINE $A8F0;

```

```

{ Misc Utility Routines }

```

```

FUNCTION GetPixel (h,v: INTEGER): BOOLEAN;    INLINE $A865;

```

```

FUNCTION Random: INTEGER;    INLINE $A861;

```

```

PROCEDURE StuffHex (thingptr: Ptr; s:Str255);    INLINE $A866;

```

```

PROCEDURE ForeColor (color: LongInt);    INLINE $A862;

```

```

PROCEDURE BackColor (color: LongInt);    INLINE $A863;

```

```

PROCEDURE ColorBit (whichBit: INTEGER);    INLINE $A864;

```

```

{ 128K ROM Routines }

```

```

PROCEDURE SeedFill(srcPtr,dstPtr: Ptr;
    srcRow,dstRow,height,words: integer;
    seedH,seedV: integer);    INLINE $A839;

```

```

PROCEDURE CalcMask(srcPtr,dstPtr: Ptr;
    srcRow,dstRow,height,words: integer);    INLINE $A838;

```

```

PROCEDURE CopyMask(srcBits,maskBits,dstBits: BitMap;
    srcRect,maskRect,dstRect: Rect);    INLINE $A817;

```

```

PROCEDURE MeasureText(count: integer; textAddr,charLocs: Ptr);
    INLINE $A837;

```




{-----}

SPEECHINTF Pascal Interface SPEECHINTF

NAME

Speech -- MacinTalk Interface

DESCRIPTION

This file contains the interface to the MacinTalk Speech Driver
For more detailed information see the MacinTalk documentation in
the Software Supplement.

MODIFICATIONS

Tom Leonard, TML Systems

Changed to "include file" format for use with MacLanguage
Series Pascal

-----}

{ Link the file Speech.Rel for the implementation of these routines
}

{\$u Speech }

CONST

noExcpsFile = ""; {signals Reader to use only basic rules}

noReader = 'noReader'; {signals SpeechOn to NOT bring in
Reader}

fullUnitT = -4000; {error code for driver unit table full}

TYPE

SpeechErr = Integer;

SpeechRecord = array[0..99] of byte; {Driver parm block, used
internally}

SpeechPointer = ^ SpeechRecord;

SpeechHandle = ^ SpeechPointer;

Sex = (Male, Female); {for future use}


```

F0Mode      = (Natural, Robotic, NoChange);
Language     = (xEnglish, French, Spanish, German, Italian);
VoiceRecord  =
    RECORD

```

```

    theSex : Sex;
    theLanguage : Language;
    theRate : Integer;
    thePitch : Integer;
    theMode : F0Mode;
    theName : Str255;
    refCon : LongInt

```

```

    END;
VoicePtr     = ^ VoiceRecord;

```

```

FUNCTION SpeechOn (ExcpsFile : Str255;

```

```

    VAR theSpeech : SpeechHandle) : SpeechErr; EXTERNAL;

```

```

PROCEDURE SpeechOff (theSpeech : SpeechHandle);

```

```

    EXTERNAL;

```

```

PROCEDURE SpeechRate (theSpeech: SpeechHandle; theRate:
    Integer); EXTERNAL;

```

```

PROCEDURE SpeechPitch (theSpeech : SpeechHandle; thePitch :
    Integer; theMode : F0Mode); EXTERNAL;

```

```

PROCEDURE SpeechSex (theSpeech : SpeechHandle;
    theSex : Sex); EXTERNAL;

```

```

FUNCTION Reader (theSpeech : SpeechHandle;

```

```

    EnglishInput : Ptr; InputLength: Longint;

```

```

    PhoneticOutput : Handle): SpeechErr; EXTERNAL;

```

```

FUNCTION MacinTalk (theSpeech : SpeechHandle;

```

```

    Phonemes: Handle) : SpeechErr; EXTERNAL;

```




{-----

TOOLINTF Pascal Interface

TOOLINTF

NAME

ToolIntf -- Toolbox Interface (7 Feb 85)

DESCRIPTION

This file contains the interface to the Macintosh Toolbox routines.

Most of the interface routines are implemented as in-line traps. The

implementation of the remaining routines is in the file ToolTraps.obj.

For more detailed information see the following sections of the "Inside Macintosh" documentation:

- Control Manager
- Desk Manager
- Dialog Manager
- Event Manager
- Font Manager
- Menu Manager
- Resource Manager
- Scrap Manager
- TextEdit
- Toolbox Utilities
- Window Manager

AUTHOR

Pascal Interfaces by Rony Sebok

Copyright 1983, 1984, 1985 Apple Computer Inc.

MODIFICATIONS

Tom Leonard, TML Systems

Changed to "include file" format for use with MacLanguage Series Pascal

Additions for the 128K ROMs are added February 1986.
DO NOT attempt to use these with the 64K ROMs.

-----}

{ Link the file ToolTraps.Rel to provide for EXTERNAL subprograms}
{ \$u ToolTraps }

CONST { Font Manager }

systemFont = 0;

applFont = 1;

NewYork = 2;

Geneva = 3;

Monaco = 4;

Venice = 5;

London = 6;

Athens = 7;

SanFran = 8;

Toronto = 9;

Cairo = 11;

LosAngeles = 12;

propFont = \$9000;

fixedFont = \$B000;

fontWid = \$ACB0;

{for Window Manager}

{window messages}

wDraw = 0;

wHit = 1;

wCalcRgns = 2;

wNew = 3;

wDispose = 4;

Pr 6 - 3

```
wGrow      = 5;  
wDrawGlcon = 6;
```

```
{types of windows}  
dialogKind = 2;  
userKind   = 8;
```

```
{window definition procedure IDs}
```

```
documentProc = 0;
```

```
dBoxProc     = 1;
```

```
plainDBox    = 2;
```

```
altDBoxProc  = 3;
```

```
noGrowDocProc = 4;
```

```
rDocProc     = 16;
```

```
{FindWindow Result Codes}
```

```
inDesk       = 0;
```

```
inMenuBar    = 1;
```

```
inSysWindow  = 2;
```

```
inContent    = 3;
```

```
inDrag       = 4;
```

```
inGrow       = 5;
```

```
inGoAway     = 6;
```

```
{defProc hit test codes}
```

```
wNoHit       = 0;
```

```
wInContent   = 1;
```

```
wInDrag      = 2;
```

```
wInGrow      = 3;
```

```
wInGoAway    = 4;
```

```
{axis constraints for DragGrayRgn call}
```

```
noConstraint = 0;
```

```
hAxisOnly    = 1;
```

```
vAxisOnly    = 2;
```



```
{for TextEdit}
```

```
teJustLeft    = 0;
teJustRight   = -1;
teJustCenter  = 1;
```

```
{for Resource Manager}
```

```
{Resource attribute byte}
```

```
resSysRef= 128; { Reference to system resource? }
resSysHeap= 64; { System or application heap? }
resPurgeable= 32; { Purgeable resource? }
resLocked= 16; { Load it in locked? }
resProtected= 8; { Protected? }
resPreload= 4; { Load in on OpenResFile? }
resChanged= 2; { Resource changed? }
mapReadOnly= 128; { Resource file read-only }
mapCompact= 64; { Compact resource file }
mapChanged= 32; { Write map out at update }
resNotFound= -192; { Resource not found }
resFNotFound= -193; { Resource file not found }
addResFailed= -194; { AddResource failed }
addRefFailed= -195; { AddReference failed }
rmvResFailed= -196; { RmveResource failed }
rmvRefFailed= -197; { RmveReference failed }
```

```
{ID's for resources provided in sysResDef}
```

```
{standard cursor definitions}
```

```
iBeamCursor = 1;    {text selection cursor}
crossCursor = 2;    {for drawing graphics}
watchCursor = 4;    {for indicating a long delay}
```


Pr 6 - 5

```
{icons}
stopIcon = 0;
notelcon = 1;
cautionIcon = 2;

{patterns}
sysPatListID = 0; {ID of PAT# which contains 38 patterns}

{for Control Manager}
{control messages}
drawCntl = 0;
testCntl = 1;
calcCRgns = 2;
initCntl = 3;
dispCntl = 4;
posCntl = 5;
thumbCntl = 6;
dragCntl = 7;
autoTrack = 8;

{FindControl Result Codes}
inButton = 10;
inCheckbox = 11;
inUpButton = 20;
inDownButton = 21;
inPageUp = 22;
inPageDown = 23;
inThumb = 129;

{control definition proc ID's}
pushButProc = 0;
checkBoxProc = 1;
radioButProc = 2;
```



```

scrollBarProc = 16;
useWFont      = 8;
{for Dialog Manager}
stdFile = 3;
userItem  = 0;
ctrlItem  = 4;
btnCtrl   = 0; { Low two bits specify what kind of control }
chkCtrl   = 1;
radCtrl   = 2;
resCtrl   = 3;
statText  = 8;   { Static text }
editText  = 16;  { Editable text }
iconItem  = 32;  { Icon item }
picItem   = 64;  { Picture item }

RECORD
itemDisable= 128; { Disable item if set }

OK = 1; { OK button is first by convention }
Cancel = 2; { Cancel button is second by convention }

{for Menu Manager}
noMark      = 0; { mark symbols for MarkItem }
commandMark = $11;
checkMark   = $12;
diamondMark = $13;
appleMark   = $14;
TextMenuProc= 0;
{ menu defProc messages }
mDrawMsg    = 0;
mChooseMsg  = 1;
mSizeMsg    = 2;

{for Scrap Manager}

```


Pr 6 - 7

```
noScrapErr = -100; {desk scrap isn't initialized}
noTypeErr = -102; {package manager}
dskInit = 2; {Disk Initializaton}
stdFile = 3; {Standard File}
flPoint = 4; {Floating-Point Arithmetic}
trFunc = 5; {Transcendental Functions}
intUtil = 6; {International Utilities}
bdConv = 7; {Binary/Decimal Conversion}
```

TYPE

```
{General Utilities}
rowLoc: Integer; { offset to offset/width table }
Int64Bit = Integer; { ascent }
RECORD Integer; { descent }
hiLong: LongInt; { leading }
loLong: LongInt; { row width of bit image / 2 }
END;
CursPtr = ^Cursor;
CursHandle = ^CursPtr;
PatPtr = ^Pattern;
PatHandle = ^PatPtr;

{ Font Manager}
FMInput =
  PACKED RECORD
family: Integer;
size: Integer;
face: Style;
needBits: Boolean;
device: Integer;
numer: Point;
denom: Point;
END;
```



```
FontRec =
  RECORD
fontType:   Integer; { font type }
firstChar:  Integer; { ASCII code of first character }
lastChar:   Integer; { ASCII code of last character }
widMax:     Integer; { maximum character width }
kernMax:    Integer; { negative of maximum character kern }
nDescent:   Integer; { negative of descent }
fRectWidth: Integer; { width of font rectangle }
```

```
fRectHeight: Integer; { height of font rectangle }
owTLoc:       Integer; { offset to offset/width table }
ascent:       Integer; { ascent }
descent:      Integer; { descent }
leading:      Integer; { leading }
rowWords:     Integer; { row width of bit image / 2 }
{ bitImage: ARRAY[1..rowWords,1..chHeight] OF Integer;
locTable:     ARRAY[firstChar..lastChar+2] OF Integer;
owTable:      ARRAY[firstChar..lastChar+2] OF Integer; }
  END;
```

```
{ 128K ROM Font Manager (for PROCEDURE FontMetrics) }
```

```
FontMetricRec =
  RECORD
ascent:      Fixed;
descent:     Fixed;
leading:     Fixed;
widMax:      Fixed;
WTabHandle: Handle
  END;
{ Event Manager }
```

```
{the Event Record is defined in OsIntf }
```



```
KeyMap = PACKED ARRAY[0..127] OF Boolean;
```

```
{for Window Manager}
```

```
WindowPtr = GrafPtr;
```

```
WindowPeek = ^WindowRecord;
```

```
ControlHandle = ^ControlPtr;
```

```
{for Control Manager}
```

```
WindowRecord =
```

```
  RECORD
```

```
  port: GrafPort;
```

```
  windowKind: Integer;
```

```
  visible: Boolean;
```

```
  hilited: Boolean;
```

```
  goAwayFlag: Boolean;
```

```
  spareFlag: Boolean;
```

```
  strucRgn: RgnHandle;
```

```
  contRgn: RgnHandle;
```

```
  updateRgn: RgnHandle;
```

```
  windowDefProc: Handle;
```

```
  dataHandle: Handle;
```

```
  titleHandle: StringHandle;
```

```
  titleWidth: Integer;
```

```
  ControlList: ControlHandle;
```

```
  nextWindow: WindowPeek;
```

```
  windowPic: PicHandle;
```

```
  refCon: LongInt;
```

```
  END;
```

```
{for TextEdit}
```

```
TERec =
```

```
  RECORD
```



```

destRect: Rect;      {Destination rectangle}
viewRect: Rect      {view rectangle}
selRect: Rect;      {Select rectangle}
lineHeight: Integer; {Current font lineheight}
fontAscent: Integer; {Current font ascent}
selPoint: Point;     {Selection point(mouseLoc)}
selStart: Integer;   {Selection start}
selEnd: Integer;     {Selection end}
active: Integer;     {<>0 if active}
wordBreak: ProcPtr;  {Word break routine}
klikLoop: ProcPtr;   {Click loop routine}
clickTime: LongInt;  {Time of first click}
clickLoc: Integer;   {Char. location of click}
caretTime: LongInt;  {Time for next caret blink}
caretState: Integer; {On/active Booleans}
just: Integer;       {fill style}
TELength: Integer;   {Length of text below}
hText: Handle;       {Handle to actual text}
recalBack: Integer;  {<>0 if recal in background}
recalLines: Integer; {Line being recal'ed}
klikStuff: Integer;  {click stuff (internal)}
crOnly: Integer;     {Set to -1 if CR line breaks only}
txFont: Integer;     {Text Font}
txFace: Style;       {Text Face}
txMode: Integer;     {Text Mode}
txSize: Integer;     {Text Size}
inPort: GrafPtr;     {Grafport}
highHook: ProcPtr;   {Highlighting hook}
caretHook: ProcPtr;  {Highlighting hook}
nLines: Integer;     {Number of lines}
lineStarts: ARRAY [0..16000] OF Integer;
                    {Actual line starts themselves}

END;

```

```

TEPtr = ^TERec;
TEHandle = ^TEPtr;

```



```
CharsHandle = ^CharsPtr;  
CharsPtr    = ^Chars;  
Chars       = PACKED ARRAY[0..32000] OF CHAR;
```

```
{for Resource Manager}
```

```
ResType=PACKED ARRAY[1..4] OF CHAR;
```

```
{for Control Manager}
```

```
ControlPtr    = ^ControlRecord;
```

```
ControlRecord =  
  PACKED RECORD
```

```
  nextControl:  ControlHandle;  
  contrlOwner:  WindowPtr;  
  contrlRect:   Rect;  
  contrlVis:    Byte;  
  contrlHilite: Byte;  
  contrlValue:  Integer;  
  contrlMin:    Integer;  
  contrlMax:    Integer;  
  contrlDefProc: Handle;  
  contrlData:   Handle;  
  contrlAction: ProcPtr;  
  contrlrfCon:  LongInt;  
  contrlTitle:  Str255;  
  END; {ControlRecord}
```

```
{for Dialog Manager}
```

```
DialogPtr=    WindowPtr;  
DialogPeek=   ^DialogRecord;
```


DialogRecord=

Record

 window:WindowRecord;

 Items:Handle;

 textH:TEHandle;

 EditField:Integer;

 EditOpen:Integer;

 ADefItem:Integer;

End;

DialogTHndl= ^DialogTPtr;

DialogTPtr= ^DialogTemplate;

DialogTemplate=

RECORD

 boundsRect:Rect;

 procID:Integer;

 visible:Boolean;

 filler1:Boolean;

 goAwayFlag:Boolean;

 filler2:Boolean;

 refCon:LongInt;

 ItemsID:Integer;

 title:Str255;

END;

StageList=

PACKED ARRAY [1..4] OF RECORD

 boldItem: 0..1;

 boxDrawn: Boolean;

 sound: 0..3;

END;

AlertTHndl= ^AlertTPtr;

AlertTPtr= ^AlertTemplate;


```

AlertTemplate=
RECORD
    boundsRect: Rect;
    itemsID: Integer;
    stages: StageList;
END;

{for Menu Manager}

MenuPtr = ^MenuInfo;
MenuHandle = ^MenuPtr;
MenuInfo =
RECORD
    menuID: Integer;
    menuWidth: Integer;
    menuHeight: Integer;
    menuProc: Handle;
    enableFlags: LongInt;
    menuData: Str255;
END;

{for Scrap Manager}

ScrapStuff =
RECORD
    scrapSize: LongInt;
    scrapHandle: Handle;
    scrapCount: Integer;
    scrapState: Integer;
    scrapName: StringPtr;
END;

pScrapStuff = ^ScrapStuff;

{General Utilities}

FUNCTION BitAnd (long1, long2: LongInt): LongInt;

```



```

FUNCTION BitOr (long1,long2: LongInt): LongInt;      INLINE $A85B;
FUNCTION BitXor (long1,long2: LongInt): LongInt;      INLINE $A859;
FUNCTION BitNot (long: LongInt): LongInt;             INLINE $A85A;
FUNCTION BitShift (long: LongInt; count: Integer): LongInt;
PROCEDURE ShiftCursor(shifted: Rect; shift: Integer);  INLINE $A85C;
FUNCTION BitTst (bytePtr: Ptr; bitNum: LongInt): Boolean;
PROCEDURE SetByte(bytePtr: Ptr; bitNum: LongInt);      INLINE $A85D;
PROCEDURE BitSet (bytePtr: Ptr; bitNum: LongInt);      INLINE $A85E;
PROCEDURE BitClr (bytePtr: Ptr; bitNum: LongInt);      INLINE $A85F;
PROCEDURE LongMul (a,b: LongInt; VAR dst: Int64Bit);
PROCEDURE IntMul (a,b: Integer; VAR dst: Int64Bit);      INLINE $A867;
FUNCTION FixMul (a,b: Fixed): Fixed;                  INLINE $A868;
FUNCTION FixRatio (numer,denom: Integer): Fixed;      INLINE $A869;
FUNCTION HiWord (x: LongInt): Integer;                INLINE $A86A;
FUNCTION LoWord (x: LongInt): Integer;                INLINE $A86B;
FUNCTION FixRound (x: Fixed): Integer;                INLINE $A86C;
PROCEDURE PackBits (VAR srcPtr,dstPtr: Ptr;
    srcBytes: Integer);                               INLINE $A8CF;
PROCEDURE UnPackBits (VAR srcPtr,dstPtr: Ptr;
    dstBytes: Integer);                               INLINE $A8D0;
FUNCTION SlopeFromAngle(angle: Integer): Fixed;       INLINE $A8BC;
FUNCTION AngleFromSlope(slope: Fixed): Integer;       INLINE $A8C4;
FUNCTION DeltaPoint(ptA,ptB: Point): LongInt;         INLINE $A94F;
FUNCTION NewString(theString:Str255): StringHandle;
                                                    INLINE $A906;
PROCEDURE SetString(theString:StringHandle; strNew: Str255);
                                                    INLINE $A907;
FUNCTION GetString(stringID: Integer): StringHandle;
                                                    INLINE $A9BA;
PROCEDURE GetIndString(VAR theString: str255; strListID: Integer;
    index: Integer);                                EXTERNAL;
FUNCTION Munger(h: Handle; offset: LongInt; ptr1: Ptr; len1:
    LongInt; ptr2: Ptr; len2: LongInt): LongInt;      INLINE $A9E0;
FUNCTION GetIcon(iconID: Integer): Handle;            INLINE $A9BB;
PROCEDURE PlotIcon(theRect: Rect; theIcon: Handle);   INLINE $A94B;
FUNCTION GetCursor(cursorID: Integer): CursHandle;    INLINE $A9B9;

```



```

FUNCTION GetPattern(patID: Integer): PatHandle;      INLINE $A9B8;
FUNCTION GetPicture(picID: Integer): PicHandle;      INLINE $A9BC;
PROCEDURE GetIndPattern(VAR thePat: Pattern; patListID: Integer;
    index: Integer);                                EXTERNAL;
PROCEDURE ShieldCursor(shieldRect: Rect; offsetPt: Point);
                                                    INLINE $A855;
PROCEDURE ScreenRes(VAR scrnHRes,scrnVRes:INTEGER);EXTERNAL;

```

{ Font Manager Routines }

```

PROCEDURE InitFonts;                                INLINE $A8FE;
PROCEDURE GetFontName(familyID: Integer; VAR theName: Str255);
                                                    INLINE $A8FF;
PROCEDURE GetFNum(theName: Str255; VAR familyID: Integer);
                                                    INLINE $A900;
PROCEDURE SetFontLock(lockFlag: Boolean);            INLINE $A903;
FUNCTION FMSwapFont(inRec: FMInput): FMOutPtr;      INLINE $A901;
FUNCTION RealFont(famID: Integer; size: Integer): Boolean;
                                                    INLINE $A902;

```

{ 128K ROM Font Manager routines }

```

PROCEDURE SetFScaleDisable(scaleDis: boolean);      INLINE $A834;
PROCEDURE FontMetrics(VAR theMetrics: FontMetricRec);
                                                    INLINE $A835;

```

{ Event Manager Routines }

```

FUNCTION EventAvail(mask:Integer;
    VAR theEvent: EventRecord): Boolean;              INLINE $A971;
FUNCTION GetNextEvent(mask:Integer;
    VAR theEvent: EventRecord): Boolean;              INLINE $A970;
FUNCTION StillDown:Boolean;                          INLINE $A973;
FUNCTION WaitMouseUp:Boolean;                        INLINE $A977;
PROCEDURE GetMouse (VAR pt: Point);                  INLINE $A972;
FUNCTION TickCount:LongInt;                          INLINE $A975;

```



```

FUNCTION Button: Boolean;                               INLINE $A974;
PROCEDURE GetKeys(VAR k: keyMap);                       INLINE $A976;
FUNCTION GetDbITime: LongInt;                             EXTERNAL;
FUNCTION GetCaretTime: LongInt;                           EXTERNAL;

{ Window Manager Routines }

PROCEDURE ClipAbove(window: WindowPeek);                INLINE $A90B;
PROCEDURE PaintOne(window: WindowPeek; clobbered: RgnHandle);
                                                         INLINE $A90C;
PROCEDURE PaintBehind(startWindow: WindowPeek; clobbered:
RgnHandle);                                             INLINE $A90D;
PROCEDURE SaveOld(window: WindowPeek);                  INLINE $A90E;
PROCEDURE DrawNew(window: WindowPeek; fUpdate: Boolean);
                                                         INLINE $A90F;
PROCEDURE CalcVis(window: WindowPeek);                  INLINE $A909;
PROCEDURE CalcVisBehind(startWindow: WindowPeek; clobbered:
  RgnHandle);                                           INLINE $A90A;
PROCEDURE ShowHide(window: WindowPtr; showFlag: Boolean);
                                                         INLINE $A908;
FUNCTION CheckUpdate(VAR theEvent: EventRecord): Boolean;
                                                         INLINE $A911;
PROCEDURE GetWMgrPort(VAR wPort: GrafPtr);              INLINE $A910;
PROCEDURE InitWindows;                                   INLINE $A912;
FUNCTION NewWindow(wStorage: Ptr;
  boundsRect: Rect;
  title: Str255;
  visible: Boolean;
  theProc: Integer;
  behind: WindowPtr;
  goAwayFlag: Boolean;
  refCon: LongInt): WindowPtr;                          INLINE $A913;
PROCEDURE DisposeWindow(theWindow: WindowPtr);          INLINE $A914;
PROCEDURE CloseWindow(theWindow: WindowPtr);            INLINE $A92D;
PROCEDURE MoveWindow (theWindow: WindowPtr; h,v: Integer;

```



```

BringToFront:Boolean);                               INLINE $A91B;
PROCEDURE SizeWindow(theWindow: WindowPtr; width, height:
    Integer; fUpdate: Boolean);                       INLINE $A91D;
FUNCTION GrowWindow (theWindow: windowPtr;
    startPt: Point; bBox: Rect): LongInt;             INLINE $A92B;
PROCEDURE DragWindow (theWindow: WindowPtr; startPt: Point;
    boundsRect: Rect);                               INLINE $A925;
PROCEDURE ShowWindow (theWindow: WindowPtr);          INLINE $A915;
PROCEDURE HideWindow (theWindow: WindowPtr);          INLINE $A916;
PROCEDURE SetWTitle (theWindow: WindowPtr; title: Str255);
                                                        INLINE $A91A;
PROCEDURE GetWTitle (theWindow: WindowPtr;
    VAR title: Str255);                               INLINE $A919;
PROCEDURE HiliteWindow (theWindow: WindowPtr;
    fHiLite: Boolean);                               INLINE $A91C;
PROCEDURE BeginUpdate (theWindow: WindowPtr);         INLINE $A922;
PROCEDURE EndUpdate (theWindow: WindowPtr);           INLINE $A923;
PROCEDURE SetWRefCon (theWindow: WindowPtr; data: LongInt);
                                                        INLINE $A918;
FUNCTION GetWRefCon (theWindow: WindowPtr): LongInt;
                                                        INLINE $A917;
PROCEDURE SetWindowPic (theWindow: WindowPtr;
    thePic: PicHandle);                               INLINE $A92E;
FUNCTION GetWindowPic (theWindow: WindowPtr): PicHandle;
                                                        INLINE $A92F;
PROCEDURE BringToFront (theWindow: WindowPtr);        INLINE $A920;
PROCEDURE SendBehind (theWindow, behind: WindowPtr);
                                                        INLINE $A921;
FUNCTION FrontWindow: WindowPtr;                      INLINE $A924;
PROCEDURE SelectWindow (theWindow: WindowPtr);        INLINE $A91F;
FUNCTION TrackGoAway(theWindow: WindowPtr; thePt: Point):
    Boolean;                                           INLINE $A91E;
PROCEDURE DrawGrowIcon (theWindow: WindowPtr);        INLINE $A904;
PROCEDURE ValidRect (goodRect: Rect);                 INLINE $A92A;
PROCEDURE ValidRgn (goodRgn: RgnHandle);              INLINE $A929;
PROCEDURE InvalRect (badRect: Rect);                 INLINE $A928;

```



```

PROCEDURE InvalRgn (badRgn: RgnHandle);          INLINE $A927;
FUNCTION FindWindow (thePoint: Point;             LONGINT;
  VAR theWindow: WindowPtr): Integer;             INLINE $A92C;
FUNCTION GetNewWindow (windowID: Integer; wStorage: Ptr;
  behind: WindowPtr): WindowPtr;                  INLINE $A9BD;
FUNCTION PinRect(theRect: Rect; thePt: Point): LongInt;
  h: TEHandle );                                  INLINE $A94E;
FUNCTION DragGrayRgn (theRgn: RgnHandle; startPt: Point;
  boundsRect, slopRect: Rect; axis: Integer;
  actionProc: ProcPtr): LongInt;                  INLINE $A905;
{ 128K ROM Window Manager Routines }

FUNCTION TextBox (window: windowPtr;
  thePt: Point;
  partCode: integer): boolean;                    INLINE $A83B;
PROCEDURE ZoomWindow(window: windowPtr;
  partCode: integer;
  front: boolean);                                INLINE $A83A;

{ TextEdit Routines }

PROCEDURE TEActivate(h: TEHandle );               INLINE $A9D8;
PROCEDURE TECalText(h: TEHandle );               INLINE $A9D0;
PROCEDURE TEClick(pt: Point; extend: Boolean; h: TEHandle );
  h: TEHandle );                                  INLINE $A9D4;
PROCEDURE TECopy(h: TEHandle );                   INLINE $A9D5;
PROCEDURE TECut(h: TEHandle );                    INLINE $A9D6;
PROCEDURE TEdoActivate(h: TEHandle );             INLINE $A9D9;
PROCEDURE TEdoDelete(h: TEHandle );               INLINE $A9D7;
PROCEDURE TEdoDispose(h: TEHandle );              INLINE $A9CD;
PROCEDURE TEdoIdle(h: TEHandle );                 INLINE $A9DA;
PROCEDURE TEdoInit;                               INLINE $A9CC;
PROCEDURE TEKey(key: CHAR; h: TEHandle );         INLINE $A9DC;
FUNCTION TENew(dest, view: Rect ): TEHandle;      INLINE $A9D2;
PROCEDURE TEPaste(h: TEHandle );                 INLINE $A9DB;

```



```

PROCEDURE TESScroll(dh, dv: Integer; h: TEHandle );      INLINE $A9DD;
PROCEDURE TESSetSelect(selStart, selEnd: LongInt;
    h: TEHandle);      INLINE $A9D1;
PROCEDURE TESSetText(inText: Ptr; textLength: LongInt;
    h: TEHandle );      INLINE $A9CF;
PROCEDURE TEInsert( inText: Ptr; textLength: LongInt;
    h: TEHandle );      INLINE $A9DE;
PROCEDURE TEUpdate( rUpdate: Rect; h: TEHandle );      INLINE $A9D3;
PROCEDURE TESSetJust( just: Integer; h: TEHandle );      INLINE $A9DF;
FUNCTION TEGetText( h: TEHandle ): CharsHandle;      INLINE $A9CB;
FUNCTION TEScrapHandle: Handle;      EXTERNAL;
FUNCTION TEGetScrapLen: LongInt;      EXTERNAL;
PROCEDURE TESSetScrapLen(length: LongInt);      EXTERNAL;
FUNCTION TEFFromScrap: OsErr;      EXTERNAL;
FUNCTION TEToScrap: OsErr;      EXTERNAL;
PROCEDURE SetWordBreak(wBrkProc: ProcPtr; hTE: TEHandle);
    EXTERNAL;
PROCEDURE SetClikLoop (clikProc: ProcPtr; hTE: TEHandle);
    EXTERNAL;
FUNCTION UniqueIDType: ResType; Integer;      INLINE
FUNCTION GetResourceIDType: ResType; ID: Integer; Handle;
    INLINE
{Box drawing utility}
PROCEDURE TextBox(inText: Ptr; textLength: LongInt;
    r: Rect; style: Integer);      INLINE $A9CE;
    INLINE $A9AT;
    INLINE $A9A2;
    INLINE $A9A3;
    INLINE $A9A4;
    INLINE $A9A5;
    INLINE $A9A6;
    INLINE $A9A7;
    INLINE $A9A8;
    INLINE $A9A9;
    INLINE $A9AA;
    INLINE $A9AB;
    INLINE $A9AC;
    INLINE $A9AD;
    INLINE $A9AE;
    INLINE $A9AF;
    INLINE $A9B0;
    INLINE $A9B1;
    INLINE $A9B2;
    INLINE $A9B3;
    INLINE $A9B4;
    INLINE $A9B5;
    INLINE $A9B6;
    INLINE $A9B7;
    INLINE $A9B8;
    INLINE $A9B9;
    INLINE $A9BA;
    INLINE $A9BB;
    INLINE $A9BC;
    INLINE $A9BD;
    INLINE $A9BE;
    INLINE $A9BF;
    INLINE $A9C0;
    INLINE $A9C1;
    INLINE $A9C2;
    INLINE $A9C3;
    INLINE $A9C4;
    INLINE $A9C5;
    INLINE $A9C6;
    INLINE $A9C7;
    INLINE $A9C8;
    INLINE $A9C9;
    INLINE $A9CA;
    INLINE $A9CB;
    INLINE $A9CC;
    INLINE $A9CD;
    INLINE $A9CE;
    INLINE $A9CF;
    INLINE $A9D0;
    INLINE $A9D1;
    INLINE $A9D2;
    INLINE $A9D3;
    INLINE $A9D4;
    INLINE $A9D5;
    INLINE $A9D6;
    INLINE $A9D7;
    INLINE $A9D8;
    INLINE $A9D9;
    INLINE $A9DA;
    INLINE $A9DB;
    INLINE $A9DC;
    INLINE $A9DD;
    INLINE $A9DE;
    INLINE $A9DF;
    INLINE $A9E0;
    INLINE $A9E1;
    INLINE $A9E2;
    INLINE $A9E3;
    INLINE $A9E4;
    INLINE $A9E5;
    INLINE $A9E6;
    INLINE $A9E7;
    INLINE $A9E8;
    INLINE $A9E9;
    INLINE $A9EA;
    INLINE $A9EB;
    INLINE $A9EC;
    INLINE $A9ED;
    INLINE $A9EE;
    INLINE $A9EF;
    INLINE $A9F0;
    INLINE $A9F1;
    INLINE $A9F2;
    INLINE $A9F3;
    INLINE $A9F4;
    INLINE $A9F5;
    INLINE $A9F6;
    INLINE $A9F7;
    INLINE $A9F8;
    INLINE $A9F9;
    INLINE $A9FA;
    INLINE $A9FB;
    INLINE $A9FC;
    INLINE $A9FD;
    INLINE $A9FE;
    INLINE $A9FF;
    { 128K ROM TextEdit Routines }

PROCEDURE TESelView (hTE: TEHandle);      INLINE $A811;
PROCEDURE TEPinScroll(dh,dv: integer; hTE: TEHandle);
    INLINE $A812;
PROCEDURE TEAutoView (auto: boolean; hTE: TEHandle);
    INLINE $A813;
    INLINE $A814;
    INLINE $A815;
    INLINE $A816;
    INLINE $A817;
    INLINE $A818;
    INLINE $A819;
    INLINE $A81A;
    INLINE $A81B;
    INLINE $A81C;
    INLINE $A81D;
    INLINE $A81E;
    INLINE $A81F;
    { Resource Manager Routines }

FUNCTION InitResources: Integer;      INLINE $A995;
PROCEDURE RsrcZoneInit;      INLINE $A996;

```



```
PROCEDURE CreateResFile(fileName: Str255);          INLINE $A9B1;
FUNCTION OpenResFile(fileName: Str255): Integer;    INLINE $A997;
PROCEDURE UseResFile(refNum: Integer);              INLINE $A998;
FUNCTION GetResFileAttrs(refNum: Integer): Integer;          INLINE $A9F6;
PROCEDURE SetResFileAttrs(refNum: Integer; attrs: Integer);  INLINE $A9F7;
PROCEDURE UpdateResFile(refNum: Integer);            INLINE $A999;
PROCEDURE CloseResFile(refNum: Integer);             INLINE $A99A;
PROCEDURE SetResPurge(install: Boolean);             INLINE $A993;
PROCEDURE SetResLoad(AutoLoad: Boolean);            INLINE $A99B;
FUNCTION CountResources(theType: ResType): Integer;      INLINE $A99C;
FUNCTION GetIndResource(theType: ResType; index: Integer): Handle;  INLINE $A99D;
FUNCTION CountTypes: Integer;                        INLINE $A99E;
PROCEDURE GetIndType(VAR theType: ResType; index: Integer);      INLINE $A99F;
FUNCTION UniqueID(theType: ResType): Integer;          INLINE $A9C1;
FUNCTION GetResource(theType: ResType; ID: Integer): Handle;     INLINE
$A9A0;
FUNCTION GetNamedResource(theType: ResType;
name: Str255): Handle;          INLINE $A9A1;
PROCEDURE LoadResource(theResource: Handle);          INLINE $A9A2;
PROCEDURE ReleaseResource(theResource: Handle);      INLINE $A9A3;
PROCEDURE DetachResource(theResource: Handle);       INLINE $A992;
PROCEDURE ChangedResource(theResource: Handle);      INLINE $A9AA;
PROCEDURE WriteResource(theResource: Handle);        INLINE $A9B0;
FUNCTION HomeResFile(theResource: Handle): Integer;   INLINE $A9A4;
FUNCTION CurResFile: Integer;                         INLINE $A994;
FUNCTION GetResAttrs(theResource: Handle): Integer;   INLINE $A9A6;
PROCEDURE SetResAttrs(theResource: Handle; attrs: Integer);  INLINE $A9A7;
```



```

PROCEDURE GetResInfo(theResource: Handle;
  VAR theID: Integer;
  VAR theType: ResType;
  VAR name: Str255);
PROCEDURE SetResInfo(theResource: Handle; theID: Integer;
  name: Str255);
PROCEDURE AddResource(theResource: Handle; theType: ResType;
  theID: Integer;
  name: Str255);
PROCEDURE AddReference(theResource: Handle; theID: Integer;
  name: Str255);
PROCEDURE RmveResource(theResource: Handle);
PROCEDURE RmveReference(theResource: Handle);
FUNCTION SizeResource(theResource: Handle): LongInt;
PROCEDURE DisposeControl (theControl: ControlHandle);
FUNCTION ResError: Integer;
PROCEDURE KillControls (theWindow: WindowPtr);
{ 128K ROM Resource Manager Routines }

FUNCTION Count1Resources(theType: ResType): Integer;
FUNCTION Get1IndResource(theType: ResType;
  index: Integer): Handle;
FUNCTION Count1Types: Integer;
PROCEDURE Get1IndType(VAR theType: ResType; index: Integer);
FUNCTION Unique1ID(theType: ResType): Integer;
FUNCTION Get1Resource(theType: ResType; ID: Integer): Handle;
FUNCTION Get1NamedResource(theType: ResType;
  name: Str255): Handle;
FUNCTION MaxSizeRsrc (theResource: Handle): LongInt;
FUNCTION RsrcMapEntry(theResource: Handle): LongInt;
FUNCTION OpenRFPPerm (filename: Str255;

```


VRefNum: integer;
 permission: byte): integer; INLINE \$A9C4;

{ Control Manager Routines }

FUNCTION NewControl(curWindow: windowPtr;
 boundsRect: Rect;
 title: Str255;
 visible: Boolean;
 value: Integer;
 min: Integer;
 max: Integer;
 contrlProc: Integer;
 refCon: LongInt): ControlHandle; INLINE \$A954;

PROCEDURE DisposeControl (theControl: ControlHandle); INLINE \$A955;

PROCEDURE KillControls (theWindow: WindowPtr); INLINE \$A956;

PROCEDURE MoveControl(theControl: ControlHandle;
 h,v: Integer); INLINE \$A959;

PROCEDURE SizeControl(theControl: ControlHandle;
 w,h: Integer); INLINE \$A95C;

PROCEDURE DragControl(theControl: ControlHandle; startPt: Point;
 bounds: Rect; slopRect: Rect; axis: Integer); INLINE \$A967;

PROCEDURE ShowControl(theControl: ControlHandle); INLINE \$A957;

PROCEDURE HideControl (theControl: ControlHandle); INLINE \$A958;

PROCEDURE SetCTitle(theControl: ControlHandle;
 title: Str255); INLINE \$A95F;

PROCEDURE GetCTitle (theControl: ControlHandle;
 VAR title: Str255); INLINE \$A95E;

PROCEDURE HiliteControl(theControl: ControlHandle;
 hiliteState: Integer); INLINE \$A95D;

PROCEDURE SetCRefCon(theControl: ControlHandle;
 data: LongInt); INLINE \$A95B;

FUNCTION GetCRefCon (theControl: ControlHandle): LongInt; INLINE \$A95A;

PROCEDURE SetCtlValue(theControl: ControlHandle;


```

    theValue: Integer);                               INLINE $A963;
FUNCTION  GetCtlValue (theControl: ControlHandle): Integer;
                                                    INLINE $A960;
FUNCTION  GetCtlMin (theControl: ControlHandle): Integer;
                                                    INLINE $A961;
FUNCTION  GetCtlMax (theControl: ControlHandle): Integer;
                                                    INLINE $A962;
PROCEDURE SetCtlMin(theControl: ControlHandle;
    theValue: Integer);                               INLINE $A964;
PROCEDURE SetCtlMax(theControl: ControlHandle;
    theValue: Integer);                               INLINE $A965;

FUNCTION  GetCtlAction(theControl: ControlHandle): ProcPtr;
                                                    INLINE $A96A;
PROCEDURE SetCtlAction(theControl: ControlHandle;
    newProc: ProcPtr);                               INLINE $A96B;

FUNCTION  TestControl(theControl: ControlHandle;
    thePt: Point): Integer;                           INLINE $A966;
FUNCTION  TrackControl(theControl: ControlHandle; thePt: Point;
    actionProc: ProcPtr): Integer;                     INLINE $A968;

FUNCTION  FindControl (thePoint: Point;
    theWindow: WindowPtr;
    VAR theControl: ControlHandle): Integer;           INLINE $A96C;
PROCEDURE DrawControls (theWindow: WindowPtr);         INLINE $A969;
FUNCTION  GetNewControl (controlID: Integer;
    owner: WindowPtr): ControlHandle;                   INLINE $A9BE;

{ 128K ROM Control Manager Routines }

PROCEDURE UpdateControls(theWindow: windowPtr;
    update: RgnHandle);                               INLINE $A953;

{ Dialog Manager Routines }

```



```

PROCEDURE InitDialogs(resumeProc: ProcPtr);          INLINE $A97B;
FUNCTION  GetNewDialog(dialogID: Integer;
           wStorage: Ptr;
           behind: WindowPtr): DialogPtr;          INLINE $A97C;
FUNCTION  NewDialog(wStorage: Ptr;
           boundsRect: Rect;
           title:      Str255;
           visible:    Boolean;
           theProc:    Integer;
           behind:     WindowPtr;
           goAwayFlag: Boolean;
           refCon:     LongInt;
           itmLstHndl: Handle): DialogPtr;          INLINE $A97D;
FUNCTION  IsDialogEvent(event: EventRecord): Boolean;  INLINE $A97F;

FUNCTION  DialogSelect( event:  EventRecord;
                        VAR theDialog: DialogPtr;
                        VAR itemHit: Integer): Boolean;  INLINE $A980;
PROCEDURE ModalDialog(filterProc: ProcPtr;
                        VAR itemHit: Integer);          INLINE $A991;
PROCEDURE DrawDialog(dialog: DialogPtr);              INLINE $A981;
PROCEDURE CloseDialog(dialog: DialogPtr);             INLINE $A982;
PROCEDURE DisposDialog(dialog: DialogPtr);            INLINE $A983;
FUNCTION  Alert(alertID: Integer;
              filterProc: ProcPtr): Integer;          INLINE $A985;
FUNCTION  StopAlert(alertID: Integer;
                   filterProc: ProcPtr): Integer;     INLINE $A986;
FUNCTION  NoteAlert(alertID: Integer;
                   filterProc: ProcPtr): Integer;     INLINE $A987;
FUNCTION  CautionAlert(alertID: Integer;
                       filterProc: ProcPtr): Integer;  INLINE $A988;
PROCEDURE CouldAlert(alertID: Integer);               INLINE $A989;
PROCEDURE FreeAlert(alertID: Integer);                INLINE $A98A;
PROCEDURE CouldDialog(DlgID: Integer);                INLINE $A979;
PROCEDURE FreeDialog(DlgID: Integer);                 INLINE $A97A;
PROCEDURE ParamText(cite0, cite1, cite2, cite3: Str255);

```



```

{ Desk Manager Routines }
PROCEDURE ErrorSound(sound: ProcPtr);          INLINE $A98B;
PROCEDURE GetDIItem(dialog: DialogPtr; itemNo: Integer;
                  VAR kind: Integer; VAR item: Handle;
                  VAR box: Rect);              INLINE $A98C;
PROCEDURE SetDIItem(dialog: DialogPtr; itemNo: Integer;
                  kind: Integer; item: Handle; box: Rect);  INLINE $A98D;
PROCEDURE SetIText(item: Handle; text: Str255);  INLINE $A98E;
PROCEDURE GetIText(item: Handle;
                  VAR text: Str255);             INLINE $A98F;
PROCEDURE SellText(dialog: DialogPtr;
                  itemNo: Integer; startSel, endSel: Integer );  INLINE $A990;
{ Menu Manager Routines }
{ Routines designed only for use in Pascal }

FUNCTION GetAlrtStage: Integer;                 EXTERNAL;
PROCEDURE ResetAlrtStage;                       EXTERNAL;

PROCEDURE DlgCut(dialog: DialogPtr);            EXTERNAL;
PROCEDURE DlgPaste(dialog: DialogPtr);         EXTERNAL;
PROCEDURE DlgCopy(dialog: DialogPtr);          EXTERNAL;
PROCEDURE DlgDelete(dialog: DialogPtr);        EXTERNAL;

PROCEDURE SetDAFont(fontNum: Integer);         EXTERNAL;

{ 128K ROM Dialog Manager Routines }

PROCEDURE HideDIItem (dialog: dialogPtr; itemNo: integer);
                                                    INLINE $A827;
PROCEDURE ShowDIItem (dialog: dialogPtr; itemNo: integer);
                                                    INLINE $A828;
PROCEDURE UpdtDialog(dialog: dialogPtr; updateRgn: rgnHandle);
                                                    INLINE $A978;
FUNCTION FindDIItem (dialog: dialogPtr;
                  thePoint: Point): integer;  INLINE $A984;

```


{ Desk Manager Routines }

FUNCTION SystemEvent(myEvent: EventRecord): Boolean;
 PROCEDURE SystemClick(theEvent: EventRecord;
 theWindow: windowPtr);
 PROCEDURE SystemTask;
 PROCEDURE SystemMenu(menuResult: LongInt);
 FUNCTION SystemEdit(editCode: Integer): Boolean;
 FUNCTION OpenDeskAcc(theAcc: Str255): Integer;
 PROCEDURE CloseDeskAcc(refNum: Integer);

{ Menu Manager Routines }

PROCEDURE InitMenus;
 FUNCTION NewMenu(menuID: Integer;
 menuTitle: Str255): menuHandle;
 FUNCTION GetMenu(rsrcID: Integer): MenuHandle;
 PROCEDURE DisposeMenu(menu: menuHandle);
 PROCEDURE AppendMenu(menu: menuHandle; data: str255);
 PROCEDURE InsertMenu (menu: MenuHandle; beforeID: Integer);
 PROCEDURE DeleteMenu (menuID: Integer);
 PROCEDURE DrawMenuBar
 PROCEDURE ClearMenuBar;
 FUNCTION GetMenuBar:Handle;
 FUNCTION GetNewMBar(menuBarID: Integer): Handle;
 PROCEDURE SetMenuBar(menuBar: Handle);
 FUNCTION MenuSelect (startPt: Point): LongInt;
 FUNCTION MenuKey (ch: CHAR): LongInt;
 PROCEDURE HiLiteMenu (menuID: Integer);


```

PROCEDURE SetItem (menu: menuHandle; item: Integer;
                  itemString: Str255);      INLINE $A947;
PROCEDURE GetItem (menu: menuHandle; item: Integer;
                  VAR itemString: Str255);  INLINE $A946;
PROCEDURE EnableItem (menu: menuHandle; item: Integer);
                                           INLINE $A939;
PROCEDURE DisableItem(menu: menuHandle; item: Integer);
                                           INLINE $A93A;
PROCEDURE CheckItem (menu: menuHandle; item: Integer;
                    checked: Boolean);      INLINE $A945;
PROCEDURE SetItemIcon(menu: menuHandle; item: Integer;
                     iconNum: Byte);       INLINE $A940;
PROCEDURE GetItemIcon(menu: menuHandle; item: Integer;
                     VAR iconNum: Byte);    INLINE $A93F;
PROCEDURE SetItemStyle(menu: menuHandle; item: Integer;
                      styleVal: Style);     INLINE $A942;
PROCEDURE GetItemStyle(menu: menuHandle; item: Integer;
                      VAR styleVal: Style);  INLINE $A941;
PROCEDURE SetItemMark(menu: menuHandle; item: Integer;
                     markChar: CHAR);       INLINE $A944;
PROCEDURE GetItemMark(menu: menuHandle; item: Integer;
                     VAR markChar: CHAR);    INLINE $A943;
PROCEDURE SetMenuFlash(flashCount: Integer);      INLINE $A94A;
PROCEDURE FlashMenuBar(menuID: Integer);          INLINE $A94C;

FUNCTION GetMHandle(menuID: Integer): menuHandle;
                                           INLINE $A949;
FUNCTION CountMItems(menu: menuHandle): Integer;
                                           INLINE $A950;
PROCEDURE AddResMenu(menu: menuHandle; theType: ResType);
                                           INLINE $A94D;
PROCEDURE InsertResMenu(menu: menuHandle; theType: ResType;
                       afterItem: Integer);      INLINE $A951;
PROCEDURE CalcMenuSize(menu: menuHandle);        INLINE $A948;

```


{ 128K ROM Menu Manager Routines }

PROCEDURE InsMenuItems(MenuHandle: Handle; itemstring: Str255; itemNum: integer);	INLINE \$A826;
PROCEDURE DelMenuItem (MenuHandle: Handle; itemNum: integer);	INLINE \$A952;

{ Scrap Manager Routines }

FUNCTION GetScrap(hDest: Handle; what: ResType; VAR offset: LongInt): LongInt;	INLINE \$A9FD;
FUNCTION InfoScrap: pScrapStuff;	INLINE \$A9F9;
FUNCTION LoadScrap: LongInt;	INLINE \$A9FB;
FUNCTION PutScrap(length: LongInt; what: ResType; source: Ptr): LongInt;	INLINE \$A9FE;
FUNCTION UnloadScrap: LongInt;	INLINE \$A9FA;
FUNCTION ZeroScrap: LongInt;	INLINE \$A9FC;

{package manager}

PROCEDURE InitAllPacks;	INLINE \$A9E6;
PROCEDURE InitPack(packID: Integer);	INLINE \$A9E5;

SONUÇ

Yapılan bu çalışmada elde edilen sonuç, beklenene çok yaklaşmıştır. Ancak Türk Dili'ne özgü bazı karakterlerin karşılıkları bulunamamıştır. Bunlar 'ı' ve 'ğ' dir. Bu çalışmada Türkçe'deki vurgulama olayı hiç dikkate alınmamıştır.

MacinTalk sürücüsü yerine yeni bir sürücü yazılması düşünülmüş, ancak yapılan araştırmada Türkçe seslerin tam bir sınıflandırması ve dökümünün olmaması ve Apple Corp.'nın Türkiye temsilcisi Bilkom A.Ş.'nin bu konuda yeterli bilgiye sahip olmaması ile birlikte; Apple Corp.'nın da MacinTalk sürücüsünün yazılımı hakkında bilgi vermemesi karşısında bu işten vazgeçilmiştir.

Sanırım daha sonra yapılacak bir çalışma ile bu çalışmanın devamı sağlanır.

KAYNAKLAR

- TML Systems
MacLanguage Series Pascal
User's Guide and Reference Manual
- TML Systems
MacLanguage Series Database Toolkit
(ISAM File Package)
User Manual
- Türk Dili Bilgisi
Prof.Dr. M.Ergin
- SmoothTalker™ Software Manual
First Byte™
- MacInTalk Development Package Version 1.0
Apple Corp.
- Maintosh Plus Kullanıcı Klavuzu
Bilkom A.Ş.



ÖZGEÇMİŞ

H.Cahit Terzioğlu 1964'te İstanbul'da doğdu. Liseyi Özel Çavuşoğlu Lisesi'nde bitirdi. 1986 yılında Yıldız Üniversitesi Bilgisayar Bilimleri Mühendisliği Bölümü'nden mezun oldu. Ocak 1987 'den bu yana mezun olduğu bölümün Yazılım Anabilim Dalı'nda Araştırma Görevlisi olarak çalışmaktadır.

