C64 TÜRKİYE

SAYI: #$04 ŞUBAT 2004

BİLGİ PAYLAŞTIKÇA ARTAR

C64 İLE KENDİ BESTENİZİ YARATIN:

DEMO MUSIC CREATOR 4.0

#CHECKSUMMER #HADES' BASIC #ROM RUTİNLERİ #KARAKTER SETİ #1541-II

(Kapak resmi *Commodore* dergisinin 3. sayısından alınmıştır.)

SAHİBİ VE BÜYÜK PATRON
İsmail "Hades" Şahin

YAZARLAR
İsmail "Hades" Şahin
Deniz Can Çelik
Bekir "Slowhand" Oğurlu

SAYFA TASARIMI
Deniz Can Çelik
+QuarkXPress

İNTERNET SİTESİ
www.geocities.com/c64turkiye

İLETİŞİM
c64turkiye@yahoo.com

TEŞEKKÜRLERİMİZLE
Çifte kavrulmuş Etibör, Uludağ gazoz, Antep fıstıklı çikolata ve bilumum diğer abur cuburlar..

EDİTÖRÜN KLAVYESİ

Uzun bir aradan sonra herkese merhaba. Yaklaşık 2 aylık bir gecikmeyle tekrar yayındayız. Bu gecikmenin elbette bir sebebi var. Kasım 2002'de Word ile hazırlanmış ilk sayımızdan bu yana en büyük değişikliğe tanık oluyorsunuz. Deniz'in çabalarıyla dergiye artık bir çeki düzen vermiş bulunuyoruz. Dergimiz artık QuarkXPress ile hazırlanıyor ve Deniz'e çabalarından dolayı kendi adıma teşekkür ediyorum.

Haberler bölümünde en son haberleri vermeye çalışsak da biz dergiyi çıkarana kadar haberler artık eskimiş oluyor. Böyle olunca Bronx 7D3 parti haberi bayatlamış oluyor. Artık herkesin bildiği, Ağustos 2003'te yapılan bir partinin Ocak 2004'te haber olarak verilmesinin doğruluğu belki tartışılır ama ülkemizde scene adına gerçekleştirilen bir faaliyeti de görmemezlikten gelemeyiz. Umarım bundan sonra daha taze haber veririz.

Derginin bu sayısı program ağırlıklı olup ayrıca son sayfalarda program döküm eklerini görebilirsiniz. Yazmış olduğum CHECKSUMMER V1.0 ile döküm ekindeki programları hatasız olarak girebilirsiniz. İkinci olarak HADES' BASIC ile C64'ün yetersiz olan Basic komutlarına yeni komutların nasıl eklendiğini öğrenebilir-siniz. ROM RUTİNLERİ ise C64'ün Romlarında bulunan ve işinize yarayabilecek bazı rutinleri tanıtıyoruz. Son olarak KARAKTER SETİ programı ile oldukça kısa bir programla C64'ün standart karakter setinin nasıl değiştirildiğini açıklamalı olarak görebilirsiniz. Assembler ile uğraşmak isteyenler için faydalı olacağına eminim.

Ayrıca C64 ile nasıl müzik yapılır öğrenmek isteyenler için Slowhand arkadaşımızın hazırladığı bir yazı işinize yarayacaktır. Ama güzel müzikler yapmak için çok çalışmak gerekmektedir. Hardware köşesinde ise bir çok kişinin işine yarayacağını düşündüğüm bir konu var.

Son olarak artık C64 Türkiye dergisinin bir sitesi var ve bu siteden eski sayıları download edebileceğiniz gibi dergide verdiğimiz programları da kısa bir süre sonra bu adreste bulabileceksiniz. İşte adresimiz: *www.geocities.com/c64turkiye*

Dergi ile ilgili her türlü fikir, eleştiri, destek vs.. için : *c64turkiye@yahoo.com* e-mail adresini kullanabilirsiniz.

Bir sonraki sayıda buluşabilmek dileği ile C64 ile kalın...

İÇİNDEKİLER:

HABERLER

SID COMPO III

Bu yıl üçüncüsü düzenlenen SID Compo, geçtiğimiz aylarda yapıldı. Türkiye'den Kürşat Karamahmutoğlu (Hydrogen/Bronx) ve bu ay bir yazıyla aramıza katılan Bekir Oğurlu (Slowhand/Independent) da yarışmaya girenler arasındaydı. C64.sk tarafından düzenlenen SID Compo III'te birbirinden güzel şarkılar yarıştı. Hatta derginin sayfaları çoğunlukla bu müzikler eşliğinde yapıldı da diyebilirim. Eğer müzikleri dinleyip de kendi bestenizi yapmak için gaza geldiyseniz, bu ayki C64 ve müzik bölümüne bir göz atın. SID Compo'daki şarkıları ise *http://www.c64.sk/index.php?content=article.php&articleid=80&id=1675* adresinde bulabilirsiniz. (Favorilerim One Hit Wonder, Illumination ve Forgotten 80's.)

RETRO REPLAY VE RR-NET

C64 için üretilen son kartuş olan Retro Replay yeniden üretilerek satışa sunuldu. Yeni modelin eskisiyle olan tek farkı siyah yerine mavi renkte olması. RR'nin yanı sıra piyasaya sürülen bir diğer ürünse RR'nin genişleme yuvasına takılan RR-Net kartı. RR-Net, C64 ile intranet bağlantısını mümkün kılıyor. Kartın üreticisi Individual Computers ise bir kampanya yaparak Retro Replay, RR-Net, Contiki işletim sistemi ve transparan kartuş kabını 100 Euro'ya satıyor. Ayrıntılı bilgi için: *http://www.jschoenfeld.de/*

HARDSID HABERLERİ

Sizlere geçen sayıda tanıttığımız HardSID Quattro PCI'ın kardeşi diyebileceğimiz HardSID PCI piyasaya sürüldü. HardSID PCI, Quattro'nun aksine tek bir SID yuvası içeriyor. Ayrıca SIDPlay1-2, CCS64, VICE, Minus/4 ve GoatTracker gibi yazılımlarla uyumlu çalışabiliyor. Diğer bir haber ise, HardSID için MacOS X sürücülerinin yazıldığı yönünde. Eğer bir HardSID kartınız varsa, artık MacOS X yüklü Mac'lerde rahatlıkla kullanabilirsiniz. Daha ayrıntılı bilgiyi HardSID'in internet sitesinde bulabilirsiniz. *http://www.hardsid.com*

INDEPENDENT KURULDU

Türkiye C64 scene'inde artık yeni bir grup var: Independent. Independent, C64 ve PC alanında çalışma yapmak amacıyla Hades, Aegis ve Slowhand tarafından geçtiğimiz ekim ayında kuruldu. Grubun internet sitesi *http://www.independentonline.org* adresinde.

YENİ BİR TÜRKÇE C64 FORUMU

Geçtiğimiz aylarda Bronx grubunun üyeleri tarafından yeni bir C64 forumu açıldı. Adı **TR-Scener** olan foruma *http://tr-scener.obsesif.info* adresinden ulaşabilirsiniz.
Ayrıca sizlere geçen ay duyurduğumuz Commodore Türk forumunun adresi de *http://www.independentonline.org/forum* olarak değişti.

IRONSTONE NEREDE?

Geçtiğimiz 2003 yazında büyük gürültü koparak C64 dünyasına giren ve hatta 3. sayımızda kapak konusu yaptığımız Ironstone'dan, geçtiğimiz aylarda hiç ses çıkmadı. Şimdilik yaz aylarında esen fırtına dindi diyebiliriz. Biz de bu konuda en az sizin kadar merak içindeyiz. Gelişmeleri (eğer olursa tabi!) önümüzdeki sayılarda okuyabilirsiniz.

METAL WARRIOR 4

Oyunseverler joystick başına! (Klasik oyun yazısı girişi.) En sonunda aklınızı başınızdan alacak bir C64 oyunu çıktı! Metal Warrior 4, piyasadaki çoğu oyunun arasından sıyrılıyor ve kendini oynatmayı başarıyor. Covert Bitops tarafından hazırlanan MW4, 2 yıllık bir yapım sürecinin ardından piyasaya sürülmüş. MW4, bir action/adventure oyunu. Senaryo ise aynı: Yine dünyayı alçak uzaylılardan kurtarıyoruz. (Zaten ne zaman rahat yüzü gördük ki?) Oyunun açıklamasını ve tam çözümünü bu sayıya yetiştiremedik. Ama önümüzdeki sayıya tamamlamayı planlıyoruz.

Bu arada ilgilenenler için MW4'ün özel versiyonu da çıkmış durumda. Özel kutusu içinde 5,25" disket, soundtrack CD'si ve kitapçığıyla beraber 30 euroya alabilirsiniz. Ancak bu özel versiyondan sadece 30 adet üretilmiş durumda. Dolayısıyla elinizi çabuk tutmanız gerek.
MW4'ü indirmek için:
http://covertbitops.c64.org/games/mw4.zip
Özel versiyon için: *http://www.quernhorst.de/atari/mw4.html*

LOTEK64 #7 VE #8 ÇIKTI

Almanca bilenler parmak kaldırsın! 3 ayda bir yayınlanan C64 dergisi Lotek64'ün 7. ve 8. sayıları çıktı. 7. sayıda Hades ile Türkiye'deki C64 scene'i hakkında yapılan bir röportaj da bulunuyor. İndirmek için: *http://www.lotek64.com*

SIDin #5

SID ile profesyonel olarak ilgilenenlere hitap eden bir dergi olan SIDin'in 5. sayısı çıktı. Bu sayının ana konusu olarak müzik motorları (music engines) seçilmiş.
http://digilander.libero.it/ice00/tsid/sidin

SCENE WORLD #9

Scene World dergisinin 9. sayısı çıktı. Bu sayıda yaklaşık 40 adet makale ve 500 KB'lık yazı bulunuyor. Ayrıca demo partiler, röportajlar ve diğer olaylar hakkında raporlar da mevcut.
http://www.sceneworld.c64.org/

VICE 1.13

Ünlü C64 emülatörü Vice'ın 1.13 nolu sürümü çıktı. Bu sürümde RR-Net desteğinin yanı sıra geliştirilmiş Catweasel Mk.3 desteği de bulunmakta.
http://viceteam.bei.t-online.de/

SIDPLAY2/W

Adam Lorentzon tarafından yazılan SidPlay2/w, Windows altında çalışan bir SID çalıcıdır. Programın beta aşaması çıktı. İndirmek için: *http://www.d.kth.se/~d93-alo/c64/spw* adresine uğrayın.

ACID64 PLAYER v1.1b

Acid64 de Windows için yazılmış bir SID çalıcı. Ancak SidPlay gibi benzerlerinden farkı, çalışmak için bir SID çipine ihtiyaç duyması... Eğer bir HardSID veya Catweasel Mk.3 kartınız varsa Acid64 ile Sid müziklerini dinleyebilirsiniz. Acid64'ün arayüzü SidPlay'den daha kullanışlı ve düzgün gözüküyor. Ancak kullanamadığımız için ne yazık ki daha ayrıntılı bilgi veremiyoruz. İlgilenenler için internet sitesi adresi: *http://www.acid64.com*

SIDDASM 1.0

Sizin de farkettiğiniz gibi bu ay SID'den gidiyoruz. Yalnız bu programı tanıtmadan geçemedim. Gufino tarafından yazılan SidDasm, SID dosyalarını disassemble edip kaynak kodlarını çıkaran yeni bir program ve sadece 40 KB (!) yer kaplıyor. Eğer SID ile ilgileniyorsanız SidDasm'yi mutlaka deneyin. Programın hem Linux, hem de Windows (adı batsın) versiyonları mevcut durumda. Ayrıca C dilinde yazılmış kaynak kodu da programın yanına eklenmiş. İndirmek için:
http://www.student.oulu.fi/~loorni/covert/tools/siddasm1.zip
http://www.student.oulu.fi/~loorni/covert/tools/siddasm1.tar.gz

BİZ BU HATAYI NASIL YAPTIK?

Geçen sayımızda verdiğimiz Back In Time 4 haberinde SID çipinin yaratıcısı olarak duyurduğumuz Martin Galway konusunda Kürşat "Hydrogen" Karamahmutoğlu'ndan bir düzeltme geldi: "SID çipinin yaratıcısının gerçek adı **Bob Yannes**." Hydrogen'e hatamızı düzelttiği için teşekkür ediyor, sizlerden de özür diliyoruz.

COMMODORE-ONE HABERLERİ

Merakla beklenen C-One projesi hızla gelişiyor. Jens Schönfeld, yaptığı açıklamada 3 yıldır süren geliştirme aşaması sonunda donanım kısmında neredeyse son aşamaya gelindiğini ve artık işin daha çok yazılım kısmıyla ilgilenildiğini belirtti. Şu anda öncelikli olarak early startup ROM'ları üzerinde çalışılıyor. Ardından da varolan FPGA tasarımlarının yeni anakarta port edilmesi geliyor.

C-One için developer'lar aranıyor! Evet, Jens Schönfeld'in yukardaki açıklamasıyla artık donanım yerine yazılım kısmına öncelik verileceği duyurulmuştu. Eğer siz de 6502 makine dili programcılığıyla ve tabii ki C-One projesiyle ilgileniyorsanız, *http://c1boot.sourceforge.net* adresindeki bilgilerle haşır neşir olduktan sonra, daha önce yaptığınız çalışmaları referans olarak *jens@schoenfeld.de* adresine yollayarak developer olmak için başvurabilirsiniz. Developer adaylarına şimdiden başarılar diliyoruz.

Kötü sayılabilecek bir haber fiyat konusunda geliyor: C-One'ın daha önce 249 euro olan fiyatı 20 euro artarak 269 euro olmuş durumda.

Jens Schönfeld'in söylediğine göre C-One'a olan talep bayağı fazlaymış. ABD'de şimdiden Software Hut, Compuquick ve Mr. Hardware Computers gibi bilgisayar mağazaları C-One satışına talip olduklarını belirtmişler. Ne diyelim, darısı Türkiye'nin başına!..

Bu arada, Jeri Ellsworth 11 Ekim 2003'te SWRAP Chicago Commodore Expo 2003 fuarına katıldı. Fuarda çekilmiş fotoğrafları *http://www.earl-ydesigns.com/InfiniteLoop/cominfo.html* adresinde bulabilirsiniz.

C-One Teknik Özellikleri (Rev.2):

- 20 MHz 65816 işlemci (6502 uyumlu)
- MonsterSID ses çipi (16 kanal stereo ses)
- SuperVIC görüntü çipi (65535 renkten 256'sını gösterebilme)
- 1GB'a kadar bilgisayar belleği (SDRAM)
- 128MB'a kadar multimedya belleği (SIMM)
- İşletim sistemleri için Compact Flash kart yuvası
- IDE arayüzü ve 3,5" sürücü konnektörü (720K/1.44/2.88)
- C64 kartuş yuvası
- 1 PCI kart yuvası (istenirse 2 yapılabilir)
- 2 Amiga 1200 uyumlu clock-port
- 2 SID çipi yuvası
- PS/2 klavye ve fare konnektörleri
- Seri ve paralel portlar

Ayrıca C-One'da FAT dosya sistemi kullanıldığı için PC'lerden dosya transferi yapabilmek de mümkün. Aşağıdaki resimde C-One'ın son halini (Rev.2 anakart) görebilirsiniz.

BRONX 7D3

Türkiye'de C64 alanında son yıllarda bir hareketlilik yaşanıyor. Bunun en belirgin örneklerinden biri de Türkiye'deki en eski C64 gruplarından Bronx tarafından düzenlenen 7Dx demo partileri. 2002 yılında 7D2 ile başlayan bu partiler, 2003 yılında da 7D3 ile devam etti. (Merak edenler için: 7d3, 2003'ün 16'lı sayı sisteminde yazılışıdır.) Hâlâ Commodore 64 ile ilgilenmeye devam edenler (sayıları eskisi kadar çok olmasa da) 7D3'e katılıp farklı alanlarda yarıştılar. Size ülkemizde yapılan tek C64, Amiga ve PC demo partisinden bahsedeceğiz.

23-24 Ağustos 2003 tarihlerinde Kadıköy/İstanbul'da gerçekleştirilen partide C64, Amiga ve PC platformlarında yapılan yarışmaya çeşitli kategorilerde 30'dan fazla katılımcı ve birçok ürün katıldı. Partinin iki gününde de bulunan ve yarışmaya katılan biri olarak yarışmacıların son dakikaya kadar çalıştıklarını ve zaman zaman yardımlaştıklarını gördüm. Kısaca partide hem rekabet hem de yardımlaşma vardı.

Aşağıda 7D3'e C64 alanında katılan ürünlerin listesini bulabilirsiniz.

GAME COMPETITION

8 Queens (Hades/INDEPENDENT,ex-BRONX)

Bu ürün parti için tam bir sürpriz oldu diyebilirim. Yıllar öncesinden aklımda kalan bir satranç probleminin C64'e uyarlanmış halidir. Aslında öylesine kendim için programlamaya başlamıştım. Bitirince partiye katılmayı düşündüm.

256 BYTE COMPETITION

Sys 2816 (Hades/INDEPENDENT,ex-BRONX)

Benim yaptığım bir çalışma olup, demo tekniklerinde ZOOMER olarak bilinen bir efektin 256 byte'ı geçmeyecek şekilde programlanmış olan bir versiyonudur. 256 byte deyip geçmeyin, çok küçük bir hafıza ile gerçekten de güzel efektler yapılabilir.

DEMO COMPETITION

Old.Skool (Vigo/BRONX)

Bu çalışmanın programlama kısmı Vigo'ya, müziği Evilman/Ex-Bronx'a ve grafik kısmı ise (SLOWHAND) Cody/Ex-Bronx'a aittir. Bu çalışmaya party intro diyebiliriz.

4 KB. DEMO COMPETITION

Geomtro (Skate/BRONX)

Bu çalışmada ise C64'ün grafik ekranında çeşitli geometrik şekillerin çizilişini görebilirsiniz.

PIXEL GRAPHIC COMPETITION

I-Royo (Hydrogen/BRONX)

Hydrogen/Bronx tarafından pixel pixel çizilen güzel bir grafik.

MUSIC COMPETITION

Sunrise (Wisdom/CRESCENT)
Astro2003 (Evilman/Ex-BRONX)
Alone (R.Bayliss/TND)
I Know What You Want (Evilman/Ex-BRONX)
Ode to Wis (Hydrogen/BRONX)

Partide C64 için en çok üretilen parçalardır. Müzikler dinlemeye değer diyebilirim.

Aşağıda partinin sonuçlarını bulabilirsiniz. Ayrıca 7D3 için yapılan tüm ürünleri *http://bronxwhq.org/downloads/7d3/7d3products.zip* adresinden indirebilirsiniz.7D3 ile ilgili daha çok resim, yazı ve görüş için Bronx'un internet sitesine (*http://www.bronxwhq.org*) gözatabilirsiniz.

Bu yıl mayıs ayında yapılması planlanan 7D4'ün ayrıntılarını yine Bronx'un internet sitesinden takip edebilirsiniz.

DEMO COMPETITION

Demo Name	Coder	Group	Platform	Point
1) Simulate	Spritus	RESIDENT	PC	8,50
2) Old.Skool	Vigo	BRONX	C64	6,80
3) Endo Intro	Endo	BRONX	PC	6,71

4 KB. DEMO COMPETITION

Demo Name	Coder	Group	Platform	Point
1) Geomtro	Skate	BRONX	C64	8,25

256 BYTE COMPETITION

---Demo Name---	Coder------	Group---------	Platform-------	Point----
1) Sys 2816	Hades	BRONX	C64	7,12
2) Starfield	Erik	RESIDENT	PC	6,70
3) Matrix	Erik	RESIDENT	PC	6,66

PIXEL GRAPHIC COMPETITION

---Gfx Name----------	Artist----------	Platform----------	Mode----------	Point-----
1) I-Royo	Hydrogen/BRONX	C64	320x200 I-fli 16 Colour	8,88
2) Nightshift	Turbo/BRONX	Amiga	320x256 Iff 64 Colour	8,35
3) Sorceror	Occult/LEGACY	Amiga	320x256 Iff 64 Colour	8,05
4) Ocw	Spritus/RESIDENT	PC	320x200 Mcga 256 Colour	6,80

RENDER GRAPHIC COMPETITION

---Gfx Name---------	Artist----------	Platform----	Point-----
1) Disaster	Czamp/Ind.	Pc	Obvius

MUSIC COMPETITION

---Msx Name----------	Composer----------	Platform----------	Point-----
1) Sunrise	Wisdom/CRESCENT	C64-NewSid	8,15
2) Astro2003	Evilman/BRONX	C64-NewSid	7,84
3) Alone	R.Bayliss/TND	C64-NewSid	7,78
4) I Know What You Want	Evilman/BRONX	C64-NewSid	7,47
5) Ode to Wis	Hydrogen/BRONX	C64-OldSid	7,15

SCRIPTING COMPETITION

--Script Name----------	Coder----------	Platform--------	Point-----
1)Slide	Spaztica/BRONX	Flash 6.0	5,94

SCRIPTING 256 BYTE COMPETITION

--Script Name----------	Coder----------	Platform--------	Point-----
1)256	Spaztica/BRONX	Flash 6.0	4,88

ILLUSTRATION COMPETITION

--Gfx Name-----	Artist----------	Technique-------	Point-----
1)Dworsk	E.Erdur	Ink & arcylic	Obvius

GAME COMPETITION

--Game Name----------	Coder----------	Platform--------	Point-----
1)8 Queens	HADES/BRONX	C64	8,50

ON THE FLY COMPETITION

--Product Name----------	Authors----------	Platform----------	Point----
1)Sunrise	Wisdom/CRESCENT	Msx C64-Sid	9,00
2)Astro 2003	Evilman/BRONX	Msx C64-Sid	8,70
3)Nightshift	Turbo/BRONX	Gfx Amiga Iff	8,50
4)Sorceror	Occult/LEGACY	Gfx Amiga Iff	8,38
5)Geomtro	Skate&Datura/BRONX	C64 4kb Demo	8,14
6)I know what you want	Evilman/BRONX	Msx C64-Sid	7,90
7)Old.Skool	Vigo,Evilman,Cody/BRONX	Demo C64	7,72
8)Sys 2816	Hades/BRONX	256 Byte-C64	7,70
9)Endo Intro	Endo,Turbo,Max/BRONX	Demo PC	7,42
10)Slide	Spaztica/BRONX	Script/Flash6	6,38
11)Mandel	Anne	Demo PC	6,06
12)256	Spaztica/BRONX	Script256/Flash6	5,76

CHECKSUMMER V1.0

Başlığa bakıp da "bu ne?" diye soranlar için hemen kısa bir açıklama yapalım. Hatırlarsanız derginin ilk sayısında MDE (makine dili editörü) isimli bir program vermiştik. Bu programla C64 için yazılmış programları belli bir formatta hatasız olarak yazabiliyordunuz. Eğer elinizde MDE formatında yazılmış bir doküman varsa bu program işinize yarıyordu.

C64 TÜRKİYE dergisinin program dökümlerini MDE ile hazırlamayı düşünüyordum. Dergiye koymak istediğim programların listesini MDE ile yazıcıdan çıkarttıktan sonra bunları WORD'de tek tek girersem derginin program dökümü hazır olacak ve gerekli açıklamaları da ekleyince işim bitecekti. Kolay gibi gözükse de aslında benim kullanacağım yöntem tam bir işkenceydi. Üstelik hata yapma ihtimali de oldukça yüksekti. Buna bir çare bulmak gerekiyordu. Öyle bir program olmalıydı ki, bu program ile benim PC üzerinde yazıp derlemiş olduğum C64 assembler programlarını MDE formatında listelemeli, kontrol toplamını hesaplamalı ve elde edilen listeyi .txt uzantılı olarak kaydetmeliydi. SLOWHAND ile yaptığım bir ICQ sohbeti sırasında bu programdan bahsettim ve 2 saat içinde istediğim özelliklerin hepsini yerine getiren bir programı DELPHI ile yazdı. Böylece dergi için döküm hazırlama işini PC ile yapabilir hale geldik. Bu dökümleri ister emülatör ile ister gerçek bir C64'te yazabilmek için gerekli olan programı ise ben hazırladım. Sonuçta CHECKSUMMER V1.0 ortaya çıktı. Fakat bu program "MDE" kadar gelişmiş bir program değil. Sadece program dökümlerindeki baytları girmeye yarıyor. MDE'deki gibi herhangi bir programı yükleyip MDE formatında liste alma özelliği, printerden çıkış alma özelliği ve hepsinden önemlisi bir program yazarken istediğiniz anda SAVE yapma ve başka bir zaman yazma işlemine kaldığınız yerden devam etme özelliği yok. Ya programı tek seferde yazıp SAVE edeceksiniz ya da aşağıdaki işlemleri yapacaksınız.

1 - Programı kaydetme işini mutlaka yeni bir satıra geçtiğiniz zaman yapın.
2 - Yeni satırdaki adresi bir yere kaydedin. Fakat sakın kaybetmeyin.
3 - RUN/STOP-RESTORE tuşlarına basarak CHECKSUMMER'dan çıkın.
4 - Aşağıdaki POKE komutlarını dikkatlice girin.

POKE 43, başlangıç adresi alçak byte
POKE 44, başlangıç adresi yüksek byte
POKE 45, bitiş adresi alçak byte
POKE 46, bitiş adresi yüksek byte

Ve diskete kayıt için

```
SAVE "programadı",8,1
```

kasete kayıt için,

```
SAVE "programadı",1,1
```

yazın. **(NOT : Adres byte'larını onlu sayı sistemine göre girmeyi unutmayın)**

Yazma işlemine kaldığınız yerden devam etmek için ise aşağıdakileri yapın:
1 - CHECKSUMMER'ı yükleyin ve çalıştırın. Hiçbir şey yapmadan RUN/STOP-RESTORE ile çıkın.
2 - Yazmaya devam edeceğiniz programı LOAD "programadı",1,1 veya LOAD "programadı",8,1 komutlarıyla yükleyin.
3 - SYS 51968 komutu ile CHECKSUMMER'ı tekrar çalıştırın.
4 - Program adını girin. Başlangıç adresi olarak yazma işlemine ara verdiğiniz sırada not aldığınız adresi girin. Bitiş adresi olarak ise program dökümündeki bitiş adresini girin.
5 - Daha sonra yazma işlemine devam edebilirsiniz.

CHECKSUMMER ilk yüklendiğinde kendisini $CB00 - $CFFF adresleri arasına transfer eder. Böylece $0801 - $9FFF ve $C000 - $CAFF arasındaki bölgeleri kullanabilirsiniz.

CHECKSUMMER formatında verdiğimiz programlar mümkün olduğu kadar $0801 adresinden başlayacaktır. Böylece yazma ve save işleminden sonra RUN komutuyla rahatça çalışacaktır.

Şimdiye kadar dergide çıkan programların checksummer dökümünü bu sayıda bulabilirsiniz. Böylece uzun uzun assembler satırlarını girmenize gerek kalmayacak. Güle güle kullanın. Eğer bir sorun çıkarsa bana bir mail atın. İşte adresler: hades@independentonline.org ve hades6510@yahoo.com.

CHECKSUMMER'ı tamamen PC üzerinde yazdım ve denedim (Gerçek C64'te hiç denemedim ama buna benzer programları zamanında yazmıştım). Aslında "F" tuşlarıyla Directory alma, LOAD, SAVE ve EXIT işlemleri de olacaktı. Tamamen zaman buldukça yaptığım için her şeyin hazır olması çok zaman alacaktı. O nedenle V1 olarak bu şekilde yaptım. Ayrıca daha gelişmiş bir şeklini tamamen PC'de yapmayı planladık. PC versiyonunda .d64 oluşturma vs. gibi özellikler de olacak. (SLOWHAND bak okuyucularımız sabırsızlanıyor. Birazcık "FASTHAND" ol. :)) CHECKSUMMER'ın assembler listesini sayfanın sonunda bulabilirsiniz. İsteyen satır satır yazabilir :) Veya http://hades6510.sitemynet.com adresinde dosyalar bölümünden .d64 dosyasına kaydedilmiş halini ve dergide çıkan programları .zip'li olarak indirebilirsiniz.

VICE emülatörde "attack disk image" seçeneğinde bir .d64 dosyası seçilmişse, Checksummer kullanarak program yazarsanız save sonunda "programadı.prg" şeklinde bir dosya oluşur. Eğer .d64 seçimi yapılmadıysa kaydedilen program, "programadı.p00" şeklinde olur. Her iki şekilde de emülatörü kullanabilirsiniz.

Şimdi herkes klavyelerinin başına ve herkese iyi çalışmalar.

```
;-----------------------------------------;
; PROGRAM ADI: CHECKSUMMER V1             ;
; PROGRAM BİTİŞ TARİHİ: 28-09-2003        ;
; PROGRAMI YAZAN: İSMAİL "HADES" ŞAHİN    ;
;-----------------------------------------;
                *=$cb00

start_prg       sei
                lda   #$4c
                sta   sw
                lda   #<irq
                ldx   #>irq
                sta   $0314
                stx   $0315
                cli
                jmp   new_prg
;-----------------------------------------;
irq             inc   $d019
sw              .byte $4c
                .word $ea31
                ldx   #$00
up_text0        lda   #$07
                sta   $d800,x
                lda   #$20
                sta   $0428,x
                inx
                cpx   #$28
                bne   up_text0
                ldx   #$00
inp_01          nop
                lda   prg_name,x
                cmp   #$40
                bcc   write
                sec
                sbc   #$40
write           sta   $0400,x
                inx
                cpx   #$0e
                bne   inp_01
                ldx   #$00
header          lda   name,x
                cmp   #$40
                bcc   wri2
                sec
                sbc   #$40
wri2            sta   $0436-$28,x
                inx
                cpx   name_len
                bne   header
                ldx   #$00
opad            lda   oper,x
                sta   $0447-$28,x
                lda   #$20
                sta   $044b-$28
                lda   oper+4,x
                sta   $044c-$28,x
                inx
                cpx   #$04
                bne   opad
                jsr   $ea87
                jmp   $ea31
;-----------------------------------------;
new_prg         lda   #$07
                jsr   $e536
                dex
                stx   $d020
                stx   $d021
                txa
spr_loop        sta   $02c0,x
                inx
                cpx   #$40
                bne   spr_loop
                lda   #$ff
                sta   $02d5
                lda   #$0b
                sta   $07f8
                lda   #<main_text
                ldy   #>main_text
                jsr   $ab1e
                clc
                ldy   #$00          ;x poz.
                ldx   #$08          ;y poz.
                jsr   $fff0
                lda   #<pre_text
                ldy   #>pre_text
                jsr   $ab1e
;-----------------------------------------;
;             isim girme bölümü
;-----------------------------------------;
                lda   #$00
                sta   name_len
                clc
                ldy   #$0e   ;x poz.
                ldx   #$0a   ;y poz.
                jsr   $fff0
                lda   #$ff
                sta   letter_cnt
isim_loop       lda   #$05
                jsr   $ffd2
                inc   letter_cnt
isim_loop_1     jsr   spr_cursor
                jsr   $ffe4
                cmp   #$0d
                beq   enter
                cmp   #$14
                beq   delete
                cmp   #$20
                bcc   isim_loop_1
isim_devam      cmp   #$60
                bcs   isim_loop_1
                jsr   $ffd2
                ldx   letter_cnt
                sta   name,x
                cpx   #$10
                bne   isim_loop
                stx   name_len
                jmp   adres_gir
;-----------------------------------------;
enter           ldx   letter_cnt
                beq   isim_loop_1
                stx   name_len
                lda   #$20
```

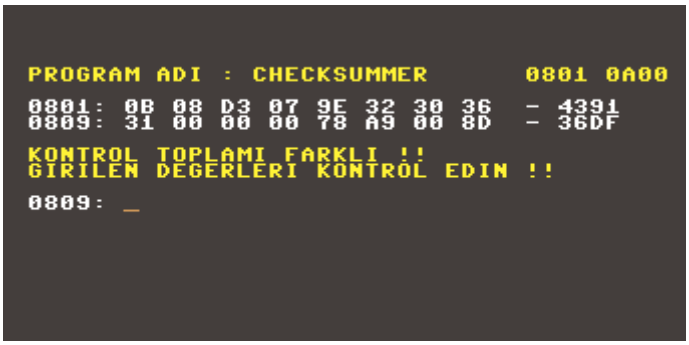
```
                    jsr    $ffd2
                    jmp    adres_gir
;-----------------------------------------;
;             silme bölümü
;-----------------------------------------;
delete              ldx    letter_cnt
                    beq    isim_loop_1
                    jsr    $ffd2
                    dec    letter_cnt
                    jmp    isim_loop_1
;-----------------------------------------;
spr_cursor          lda    $d3
                    cmp    #$1d
                    bcc    no_extra
                    ldx    #$01
                    .byte  $2c
no_extra            ldx    #$00
                    stx    $d010
                    asl
                    asl
                    asl
                    clc
                    adc    #$18
                    sta    $d000
                    lda    $d6
                    asl
                    asl
                    asl
                    clc
                    adc    #$32
                    sta    $d001
                    inc    $d027
                    lda    #$01
                    sta    $d015
                    rts
;-----------------------------------------;
;             adres girme bölümü
;-----------------------------------------;
adres_gir           clc
                    ldy    #$0f         ;x poz.
                    ldx    #$0b         ;y poz.
                    jsr    $fff0
                    jsr    key_in
                    sta    $fc
;başlangıç adresi hi byte
                    sta    start_hi
                    jsr    key_in
                    sta    $fb
;başlangıç adresi lo byte
                    sta    start_lo
                    clc
                    ldy    #$0f         ;x poz.
                    ldx    #$0c         ;y poz.
                    jsr    $fff0
                    jsr    key_in
                    sta    $fe
;bitiş adresi hi byte
                    jsr    key_in
                    sta    $fd
;bitiş adresi lo byte
                    lda    #$00
                    sta    $d015
;-----------------------------------------;
; başlangış ve bitiş adresleri kontrolü
;-----------------------------------------;
                    lda    $fe
                    cmp    $fc
                    bcc    error        ;small
                    bne    devam_yes
                    lda    $fd
                    cmp    $fb
                    beq    error        ;equal
                    bcc    error        ;small
;-----------------------------------------;
;             hata yoksa devam et
;-----------------------------------------;
devam_yes           lda    #<yes_no
                    ldy    #>yes_no
                    jsr    $ab1e
evet_no             jsr    $ffe4
                    cmp    #$45
                    beq    devam_
                    cmp    #$48
                    bne    evet_no
                    jmp    new_prg   ;start_prg
devam_              jmp    input_bytes
;-----------------------------------------;
;  hata mesajı yazdırma ve tuş kontrolu
;-----------------------------------------;
error               lda    #<error_mes
                    ldy    #>error_mes
                    jsr    $ab1e
space_key           jsr    $ffe4
                    cmp    #$20
                    bne    space_key
                    jmp    new_prg
;-----------------------------------------;
key_in              jsr    hex_key
                    asl
                    asl
                    asl
                    asl
                    sta    temp
                    jsr    hex_key
                    ora    temp
                    rts
;-----------------------------------------;
hex_key             jsr    spr_cursor
                    jsr    $ffe4
hex_sys             cmp    #$30
                    bcc    hex_key
                    cmp    #$3a
                    bcs    hex_letter
                    jsr    $ffd2
                    sec
                    sbc    #$30
                    rts
hex_letter          cmp    #$41
                    bcc    hex_key
                    cmp    #$47
                    bcs    hex_key
                    jsr    $ffd2
```

```
                    sec
                    sbc    #$37
                    rts
;-----------------------------------------;
convert             pha
                    lsr
                    lsr
                    lsr
                    lsr
                    jsr    convert0
                    sta    temp0
convert1            pla
                    and    #$0f
convert0            cmp    #$0a
                    bcs    hexharf
                    clc
                    adc    #$39
hexharf             sec
                    sbc    #$09
                    sta    temp1
                    rts
;-----------------------------------------;
convex              jsr    convert
                    lda    temp0
                    sta    oper,x
                    inx
                    lda    temp1
                    sta    oper,x
                    inx
                    rts
;-----------------------------------------;
input_bytes         ldx    #$00
                    lda    $fc       ;start hi
                    jsr    convex
                    lda    $fb       ;start lo
                    jsr    convex
                    lda    $fe       ;finish hi
                    jsr    convex
                    lda    $fd       ;finish lo
                    jsr    convex
                    lda    #$2c
                    sta    sw
                    jsr    $e544
                    dex
                    lda    #$0d
                    jsr    $ffd2
                    lda    #$05
                    jsr    $ffd2
check_1             lda    #$0d
                    jsr    $ffd2
                    ldy    #$00
                    sty    byte_cnt
                    jsr    start_adr
                    lda    #$3a
                    jsr    $ffd2
                    lda    #$20
                    jsr    $ffd2
;-----------------------------------------;
;    bir satırdaki baytların girilmesi
;-----------------------------------------;
one_line            jsr    key_in
                    ldy    byte_cnt
                    sta    ($fb),y
                    lda    #$20
                    jsr    $ffd2
                    inc    byte_cnt
                    ldy    byte_cnt
                    cpy    #$08
                    bne    one_line
                    jsr    checksum
;kontrol toplamı burada hesaplanıyor
;-----------------------------------------;
;kullanıcının kontrol toplamı girdiği yer
;-----------------------------------------;
                    jsr    key_in
                    sta    $05
                    jsr    key_in
                    sta    $04
;-----------------------------------------;
;      kontrol toplamları karşılaştırma
;      önce yüksek byte'ı karşılaştır
;-----------------------------------------;
                    lda    $05
                    cmp    $03
                    bne    check_err
;-----------------------------------------;
;      sonra alçak byte'ı karşılaştır
;-----------------------------------------;
                    lda    $04
                    cmp    $02
                    bne    check_err
;-----------------------------------------;
;      programın sonuna gelindi mi?
;-----------------------------------------;
;      önce alçak baytı karşılaştır
;-----------------------------------------;
                    lda    $fb
                    clc
                    adc    #$08
                    sta    $fb
                    bcc    no_hi_adr
                    inc    $fc
no_hi_adr           lda    $fb
                    cmp    $fd
                    bne    check_1
;-----------------------------------------;
;      sonra yüksek baytı karşılaştır
;-----------------------------------------;
                    lda    $fc
                    cmp    $fe
                    bne    check_1
                    jsr    end_of
                    jmp    save
;-----------------------------------------;
check_err           lda    #<err_mes
                    ldy    #>err_mes
                    jsr    $ab1e
                    jmp    check_1
;-----------------------------------------;
;   program sonunda yapılacak olan işler
;-----------------------------------------;
end_of              lda    #$00
```

```
                    sta    $d015
                    sei
                    lda    #$31
                    ldx    #$ea
                    sta    $0314
                    stx    $0315
                    cli
                    rts
;-----------------------------------------;
save                lda    #<save_msg
                    ldy    #>save_msg
                    jsr    $ab1e
save_key            jsr    $ffe4
                    cmp    #$45
                    beq    cihaz
                    cmp    #$48
                    bne    save_key
                    jsr    end_of
                    jmp    $a474
;-----------------------------------------;
cihaz               lda    #<kas_dis
                    ldy    #>kas_dis
                    jsr    $ab1e
kd_key              jsr    $ffe4
                    cmp    #$4b
                    beq    kaset
                    cmp    #$44
                    bne    kd_key
;-----------------------------------------;
;      SETLFS rutini için hazırlık
;-----------------------------------------;
disket              lda    #$08
                    .byte  $2c
kaset               lda    #$01
                    ldx    #$08
                    ldy    #$ff
                    jsr    $ffba
;-----------------------------------------;
;      SETNAM rutini için hazırlık
;-----------------------------------------;
                    lda    name_len
                    ldx    #<name
                    ldy    #>name
                    jsr    $ffbd
;-----------------------------------------;
;      SAVE rutini için hazırlık
;-----------------------------------------;
                    lda    start_lo
                    ldx    start_hi
                    sta    $fb
                    stx    $fc
                    lda    #$fb
                    ldx    $fd       ;finish lo
                    ldy    $fe       ;finish hi
                    jsr    $ffd8
                    jmp    $a474
;-----------------------------------------;
start_adr           lda    $fc
                    jsr    memtohex
                    lda    $fb
                    jsr    memtohex
                    rts
;-----------------------------------------;
memtohex            pha
                    lsr
                    lsr
                    lsr
                    lsr
                    jsr    nibble
                    pla
                    and    #$0f
nibble              cmp    #$0a
                    bcc    number
                    clc
                    adc    #$07
number              clc
                    adc    #$30
                    jsr    $ffd2
                    rts
;-----------------------------------------;
checksum            lda    #$00
                    sta    $02
                    sta    $03
                    lda    #<linechar1
                    ldy    #>linechar1
                    jsr    $ab1e
                    ldy    #$00
next_byte           ldx    multiple,y
read_it             lda    ($fb),y
                    clc
                    adc    $02
                    sta    $02
                    bcc    noinc03
                    inc    $03
noinc03             dex
                    bne    read_it
                    iny
                    cpy    #$08
                    bne    next_byte
                    rts
;-----------------------------------------;
pre_text            .byte 13,13
prg_name            .text  "program adi : "
                    .byte  13
                    .text  "baslama adr.: "
                    .byte  5,36
                    .byte  13,158
                    .text  "bitis adresi: "
                    .byte  5,36
                    .byte  13,0
yes_no              .byte  13,13
                    .text  "emin misiniz ?
              (e/h)"
                    .byte  0
error_mes           .byte  13,13,13
                    .text  "adres hatasi!!"
space_mes           .byte  13
                    .text  "tekrar icin 'space'
              tusuna basin !!"
                    .byte  13,0
err_mes             .byte  158,13,13
                    .text  "kontrol toplami
```

```
                farkli !!"
                       .byte  13
                       .text  "girilen degerleri
                kontrol edin !!"
                       .byte  13,5,0
save_msg               .byte  13,13
                       .text  "save ? (e/h)"
                       .byte 13,0
kas_dis                .byte  13
                       .text  "kaset/disket ?
                (k/d)"
                       .byte  13,0
main_text              .byte  $9f
                       .text  "            check
                summer  v1.0"
                       .byte  13,13
                       .text  "        (c)2003
                hades/independent"
                       .byte  $9e,0
;-----------------------------------------;
letter_cnt             .byte  0
byte_cnt               .byte  0
start_lo               .byte  0
start_hi               .byte  0
name_len               .byte  0
name                   .text  "                  "
;-----------------------------------------;
                       *=name+17
linechar1              .byte  32,45,32,5,0
oper                   .byte  0,0,0,0,0,0,0,0
temp                   .byte  0
multiple               .byte
17,19,23,29,31,37,41,59
temp0                  .byte  0
temp1                  .byte  0
end_of_prg             .byte  0
;---------------------------------------;
                       .end
```

Ve bunlar da CHECKSUMMER'in ekran görüntüleri:



HADES' BASIC

Yeni bir konu ile herkese merhaba. Başlığa bakıp BASIC öğreteceğimizi düşünmeyin. Bildiğiniz gibi -bilmiyorsanız da öğrenmiş oluyorsunuz- C64 Guinnes rekorlar kitabına giren bir ev bilgisayarıdır ve 20 senedir 2-3 revizyon dışında hiçbir değişikliğe uğramadı. Ama ne hikmetse 20 sene sonra bile hala deli gibi uğraşanlar var, ülkemizde BRONX'un 7D2, 7D3 gibi partileri haricinde hiçbir faaliyet olmasa da yurt dışında, Avrupa'da sürekli olarak partiler, yarışmalar düzenlenmektedir. (www.c64.sk adresinde haberleri bulabilirsiniz.)

Her neyse konuyu fazla dağıtmadan devam edelim. C64'ün çıktığı yıllarda ev bilgisayarları olarak birçok model vardı. Bazıları C64'ten gerçekten iyiydi. Ama 5-10 sene içinde çoğu piyasadan silindi. Kimisi pahalıydı, kimisi yeterince ilgi görmedi, kimisinin program desteği sorunu vardı vs.vs..
C64 aslında iyi bir makinaydı, zamanına göre çok güzel müzikleri ve grafikleri vardı. Bir düşünün, takvimler 1982'yi gösterirken bir bilgisayar piyasaya çıkıyor ve birkaç sene sonra o bilgisayar için yüzlerce oyun ve program üretilmiş durumda. Oyun müzikleri ve grafikleri insanı mest ediyordu. (O yıllarda PC -primitive calculator- ler siyah beyaz ekranda çalışan, EGA, CGA bilmemne modlu ekran seçen, iğrenç ötesi bip bip sesler çıkaran neredeyse sadece elektronik daktilo niyetine kullanılan makinalardı. Üstelik fiyatları da çok yüksekti.)

Konumuza geri dönersek C64'ün bu kadar tutulmasının sebebi neydi? Programlaması acaba çok mu kolaydı? Yoksa özel çipleri sayesinde birçok şeyin rahatça yapılabilir olması mıydı? Bence hem evet, hem hayır. Evet olan kısmı özel çipleri ve programlama teknikleri sayesinde bir zamanlar kimsenin -hatta C64'ü tasarlayan mühendislerin bile- aklına gelmeyen uçuk-kaçık demo efektlerinin yapılabilmesiydi. Hayır olan kısmı ise C64'de program yazmanın biraz zor olması daha doğrusu BASIC'in yetersiz olmasıdır. Bence programcılar C64 BASIC V2'yi yetersiz buldukları için assembler ile harikalar yarattılar. Gerçekten de C64'ün V2 BASIC'i çok yetersizdir. C64'ün BASIC'i **Micro(p)soft** tarafından hazırlandığı için yetersizdir. Diğer ev bilgisayarlarında ekran rengini değiştirmek için BORDER, COLOR gibi komutlar varken C64'te bu iş için POKE 53280, renkkodu komutu kullanmanız gerekmektedir. Mesela C64'ün SPRITE dediğimiz grafik özelliğini kullanmanız için -yani ekranda sadece bir sprite göstermek için- en az 4 adet POKE komutu kullanmanız gerekiyor. Üstelik adresleri de bilmeniz gerekiyor. Önce sprite'ın şekil datalarını uygun adreslere POKE etmeniz gerekiyor. Yani istediğiniz adresleri kullanamazsınız. Sonra sprite'ın X ve Y koordinatları için 2 adet, sprite pointer'i için 1 adet ve en son olarak sprite'ı görünür yapmak için 1 adet POKE komutuna ihtiyacınız vardır. C64'ün BASIC'i bu kadar yetersiz olmasına rağmen işletim sistemini tasarlayanlar o kadar esnek tasarlamışlar ki, RAM'de bulunan VECTOR adresler sayesinde BASIC V2'nin yetersizliği sorun olmaktan çıkmıştır. Türkçe COMMODORE dergisinin MART 1986 tarihli ilk sayısında SIMONS' BASIC isimli bir programdan bahsediliyordu ve o programı yazan kişinin (DAVID SIMONS) henüz 14 yaşındayken yazmış olduğu bildiriliyordu. Bu program sayesinde C64'e 100'den fazla yeni BASIC komutu eklenmiş oluyordu.

Bu kadar edebiyattan sonra ne yapacağımızı anlatalım artık. SIMONS' BASIC kadar olmasa da C64'ümüze 3-5 tane komut ekleyeceğiz. Mümkün olduğu kadar basit bir şekilde anlatacağım. (Elimdeki 06-12-2000 tarihli bir dökümana göre - HADES' BASIC olarak isim vermişim- C64'e 20 adet yeni komut eklemişim)

C64'e komut eklemenin birkaç yolu vardır:

1 - SYS komutuna parametreler ekleyerek. Mesela ekran rengini değiştirmek için POKE 53280, renk yerine SYS 49152, renk kodu gibi geliştirilmiş bir SYS komutu kullanabilirsiniz.

2 - Bir zamanlar moda olan PINKY TURBO isimli teyp için turbo load / save programındaki gibi özel bir karakter tanımlayıp mesela renk değiştirme işlemi yapmak. (PINKY TURBO aktifken hafızaya turbosuz yüklenmiş fakat henüz çalıştırmamış olduğunuz bir programı "S" komutu ile kasete turbolu olarak kaydedebilir, "L" ile kasetten bilgisayara yükleyebilir, "V" ile de VERİFY işlemi yapabiliyordunuz. Burada "" karakteri yeni bir komut olarak işletim sistemine eklenmişti.)
"C, renk kodu" gibi bir komutla renk değiştirme yapılabilir.

3 - Tamamen yeni bir komut ismi tanımlayıp işinizi yapmak. Mesele "POKE 53280, renk kodu" yerine "BORDER renk kodu" şeklinde bir komut tanımlamak.

4 - Benim bilmediğim başka bir şekilde komut eklemek :)

Bu bölümde 3. maddedeki biçimde yeni komutlarımızı ekleyeceğiz.

3. yöntemde C64'e yeni BASIC eklemek için en basit yol $0308-$0309 adreslerinde bulunan IGONE olarak bilinen BASIC KARAKTER GÖNDERİMİ vektörünü değiştirmektir. Bu vektörü değiştirince işletim sistemi normalde gitmesi gereken $A7E4 adresi yerine sizin yeni komutları değerlendirecek rutininize gider. Eğer yeni komut varsa işletilir, yoksa işletim sistemi $A7E4 adresine sıçrayıp eski işine devam eder. Şimdi hep beraber ekran silme işini yapacak "CLS" yani CLear Screen ve birkaç tane komutu C64'e kazandıralım. Benim göstereceğim yöntemde komut sıkıştırma gibi bir özellik yoktur. Yani normal komutlarda olduğu gibi bir komut için tek bir bayt kullanılmıyor. Açıklamak gerekirse siz C64'te program yazarken PRINT komutunu kullandığınızda hafızaya P,R,I,N,T harfleri değil sadece $99 sayısı kaydedilir. Yani komutlar "TOKEN" olarak saklanır. Böylece hafızadan tasarruf edilir. RUN komutu ile program normal çalışır. LIST komutu verince hafızadaki her bayt tek tek değerlendirilir ve hangisinin komut kodu hangisinin normal text olduğu çözümlenir. Örnekte olduğu gibi $99 rakamı Basic yorumlayıcı tarafından PRINT şeklinde ekrana gönderilir. İşte bu sıkıştırma özelliği benim vereceğim komutlarda olmayacaktır. Fakat yeni komutlarımızda normal bir komutun harfleri geçiyorsa -mesela yeni komutlarımızdan olan BORDER komutunda OR komutu yer almaktadır- yeni komut B,O,R,D,E,R olarak değil B, OR komutunun kodu, D,E,R olarak hafızada tutulmaktadır.

Komut sıkıştırma özelliğini kullanmak için programda "ICRNCH", "IQPLOP" ve "IEVAL" isimleriyle bilinen "BASIC TEXT KODLAMA", "BASIC TEXT LIST" ve "BASIC TOKEN DEĞERLENDİRME" vektörlerini de değiştirip uygun programları yazmak gerekmektedir.
Bu işle bir ara uğraştım ama devam ettirmedim. Aslında 3-5 komut için uğraşmaya değmez. Ama isteyen uğraşabilir. (İnternetten bulabileceğiniz SIMONS' BASIC programını inceleyebilirsiniz)

ÖNEMLİ NOT 1: Bu programı PC üzerinde VİCE emülatörü ile kullanacaksanız VİCE'da kartuş eklentisi olmamalıdır. Gerçek C64'te hatasız çalışan programı emülatör üzerinde çalıştırana kadar canım çıktı desem yeridir.

```
;-----------------------------------------;
;       C64 İÇİN YENİ BASIC KOMUTLARI     ;
;  C64 TÜRKİYE SAYI 4 İÇİN HAZIRLANMIŞTIR ;
;        12-TEMMUZ-2003     HADES         ;
;       C64ASM V1.1  ILE YAZILMIŞTIR      ;
;-----------------------------------------;
              *      = $C000
comset =cls

              lda    #<yeni
              ldx    #>yeni
              sta    $0308
              stx    $0309
              rts
;-----------------------------------------;
```

Buraya kadar olan kısımda programımızın başlangıcını $C000 yani 49152 olarak ayarladık ve yeni komutumuzu değerlendirecek kısmı "yeni" etiketi kullanarak vektörümüze yükledik. Böylece klavyeden bir komut girdiğimizde veya yazmış olduğumuz bir programı çalıştırınca önce bizim komut yorumlayıcımız devreye girecek, daha sonra biz kontrolü C64'ün kendi komut yorumlayıcısına bırakacağız.

```
;-----------------------------------------;
yeni          lda    $7a
              ldx    $7b
              sta    $fb
              stx    $fc
;-----------------------------------------;
```

Yukarıda $0073 nolu adresteki CHRGET isimli rutinde bulunan ve program içindeki baytların adresinin tutulduğu iki adresin $7a ve $7b değerlerini okuyup daha sonra kullanmak için saklıyoruz

```
;-----------------------------------------;
              lda    #<comset
              ldx    #>comset
              sta    $fd
              stx    $fe
;-----------------------------------------;
```

Üstteki satırlarda ise yeni komutlarımızın isimlerinin bulunduğu bölgenin başlangıç adresini öğrenip $fd ve $fe adreslerine kopyalıyoruz.

```
;-----------------------------------------;
              ldx    #$00
              ldy    #$00
loop0         jsr    $0073
              sta    $02
loop1         lda    ($fd),y
              cmp    #$02
              bne    nextcom
              iny
              tya
              cmp    comlen,x
              bne    loop0
;-----------------------------------------;
```

Geldik en önemli kısma. Bu bölümde klavyeden girdiğimiz veya yazdığımız program içindeki komutların yeni komut mu yoksa eski komut mu olduğunu öğreniyoruz. Nasıl oluyor anlayalım.

X registerini yeni komutların sayıcısı olarak, Y registerini de yeni komutun harf sayıcısı olarak kullanmak üzere sıfırlıyoruz. JSR komutu ile program içindeki veya klavyeden girilen o anki karakteri okuyoruz ve saklıyoruz. Sonra bizim yeni komutların başlangıç adresinin tutulduğu $fd ve $fe adreslerini kullanarak komut tablomuzdan o anki karakteri alıyoruz ve daha önce sakladığımız değerle karşılaştırıyoruz. Karşılaştırma sonucu aynı ise karakter sayacını 1 arttırıp AKÜ'ye kopyalıyoruz. Aküdeki değeri X registerini kullanarak komut uzunlukları tablosundaki o anki değerle karşılaştırıyoruz. Eğer eşit değilse klavyeden girilen veya program içindeki bir sonraki baytı almak için "LOOP0" etiketli yere geri dönüyoruz. Eğer karşılaştırma sonucu eşitse bu kez "EXEC" rutini devreye giriyor. Fakat daha önce komut tablosu ile yapılan karşılaştırma sonucunda bir farklılık varsa bu kez "NEXTCOM" etiketli yere sıçrıyoruz ve bir sonraki komut için hesaplamalar yapıyoruz.

```
;-----------------------------------------;
exec          txa
              asl
              tax
              lda    comexe+1,x
              pha
              lda    comexe,x
              pha
              rts
;-----------------------------------------;
```

Bu bölümde eğer yeni bir komut olduğu anlaşılmışsa bu komutun çalışması sağlanıyor. Bu ise komut sayacı olarak kullandığımız X registerini önce Akü'ye yükleyip 2 ile çarpıyoruz tekrar X'e kopyalıyoruz. Sonra komut çalıştırma adreslerinin tablosundan bulunan komutun çalışma adresi HIGH BYTE, LOW BYTE olarak yığına atılıyor. Böylece bilgisayar en sondaki "RTS" komutunu işlediği anda "PROGRAM COUNTER" de yeni komutun adresi bulunduğu için yeni komutu çalıştırıyor.

```
;-----------------------------------------;
nextcom       lda    comlen,x
              clc
              adc    $fd
              sta    $fd
              bcc    next0
              inc    $fe
```

```
next0       inx
            cpx    commax
            bne    loop1
            lda    $fb
            ldx    $fc
            sta    $7a
            stx    $7b
            jmp    $a7e4
;-----------------------------------------;
```

Bu bölümde ise komut tablosundaki bir sonraki yeni komutun adının adresi hesaplanarak bütün kontrol işlemine geri dönülüyor. Bunun için bir önceki komutun adının uzunluğu komut tablosunun başlangıç adresinin tutulduğu $fd adresindeki değerle toplanıyor ve geri yazılıyor. Sonra ise komut sayıcısı olan X registeri 1 arttırılıyor ve yeni komutların toplam sayısı olup olmadığı kontrol ediliyor. Eğer eşit değilse "LOOP1" etiketli yere geri dönüp karşılaştırma işlemlerini tekrar yapıyoruz.

Komut sayıcısı maximum komut sayısına eşit olduğu halde yeni komut bulunamadıysa artık kontrolü eski komut yorumlayıcısına bırakmak üzere daha önceden kopyalamış olduğumuz ve $fb ve $fc adreslerindeki değerleri $7a ve $7b adreslerine geri yazıyoruz. En son olarak JMP $A7E4 ile komut yorumlayıcısına benim işim bitti sıra sende diyoruz. Eğer standart komut değilse "SYNTAX ERROR" mesajı ekrana çıkar ve programın çalışması durur.

```
;-----------------------------------------;
;      YENİ KOMUTLAR BURADAN BAŞLIYOR     ;
;-----------------------------------------;
```

CLS komutunu normalde ekran silme için düşünmüştük. Fakat birkaç komut eklemeyle CLS komutunu hem ekran silme hem de ekrandan istediğimiz satırı silmek için kullanacağız. Sadece CLS yazarsak ekran tamamen silinir. Eğer CLS ekran satır numarası şekilde verirsek sadece o satır silinir.

```
;-----------------------------------------;
cls         jsr    $73
; Komuttan sonra sayı var mı ?
            beq    all
; Yoksa ekran tamamen silinsin
            jsr    $b79e
; Evet var ve sayıyı al
            cpx    #$01
; Sayı "1" den
            bcc    ilerr
; küçükse hata mesajı yazdır.
            cpx    #$1b
; "26" dan büyük veya eşitse
            bcs    ilerr
; hata mesajı yazdır.
            dex
; x registerini 1 azalt.
            jsr    $e9ff
; ROM'daki satır silme rutinini çağır.
            jmp    $a7ae
; Bir başka komut için devam et
all         jsr    $e544
; ROM'daki ekran silme rutinini çağır.
            jmp    $a7ae
; Bir başka komut için devam et
ilerr       jmp    $b248
; "ILLEGAL QUANTITY ERROR" mesajı yazdır.
;-----------------------------------------;
border jsr  $b79b
; Komuttan sonraki ifadeyi değerlendir.
            stx    $d020
; Sonucu VIC'in kenar rengi adresine yaz.
            jmp    $a7ae
; Bir başka komut için devam et.
;-----------------------------------------;
ink         jsr    $b79b
; Komuttan sonraki ifadeyi değerlendir.
            stx    $0286
; Sonucu karakter rengi adresine yaz.
            jmp    $a7ae
; Bir başka komut için devam et.
;-----------------------------------------;
paper       jsr    $b79b
; Komuttan sonraki ifadeyi değerlendir.
            stx    $d021
; Sonucu VIC'in sayfa rengi adresine yaz.
            jmp    $a7ae
; Başka bir komut için devam et.
comexe .word   cls-1
            .word   border-1
            .word   ink-1
            .word   paper-1
;-----------------------------------------;
cls         .text   "cls" ; "CLS"
;-----------------------------------------;
border .text   "b"          ; "B"
            .byte   $b0
; "OR" komutunun kodu
            .text   "der" ; "DER"
;-----------------------------------------;
ink         .text   "ink" ; "INK"
;-----------------------------------------;
paper       .text   "paper"     ; "PAPER"
;-----------------------------------------;
commax .byte   4
; max komut sayısı
comlen .byte   3,5,3,5
; komut uzunlukları
;=========================================
```

ÖNEMLİ NOT 2 : Programa yeni bir komut eklediğinizde komutlar "comexe" bölümündeki sıra nasılsa "comset" bölümündeki sıra da aynı olmalıdır. Aynı şekilde "comlen" değerleri de doğru sırada olmalıdır.

ÖNEMLİ NOT 3 : Yeni komutları SYS 49152 komutuyla aktif hale getirdikten sonra NEW komutu vermeniz gerekmektedir.

İşte yeni komutlar kullanarak yazılan ilk programımız ve ekran görüntüsü.

```
10 CLS : REM "PRINT CHR$(147)" YERİNE
20 FOR F=0 TO 15
30 BORDER F : REM "POKE 53280,F" YERİNE
40 NEXT : GOTO 20
```

ROM RUTİNLERİ - 1

Herkese merhaba. Yeni bir bölümle daha karşınızdayız. Umarım işinize yarar. Bu bölümde sizi uzun uzun bazı rutinleri yazmaktan kurtaracak ROM adresleri ve bunlarla ilgili kısa kısa rutinler vereceğim.

C64 mühendisleri - assemblerci olanlar - bizler için zamanında birçok hazır rutin hazırlamış olup bize de bunları kullanmak düşer. İlk adresimiz ekran silme işini yerine getiren **$E544**.

ADRES: $E544
GÖREVİ: EKRANI SİLER
KLAVYE KARŞILIĞI: SHIFT+CLR/HOME

Aslında bu adres C64'ün ilk açılışta bazı işlemleri yaparken uğradığı $E518 adresindeki rutinin bir parçası olup, "THE ANATOMY OF A C64" kitabında "CLEAR SCREEN" olarak tanımlanmıştır. Bu adres sizi bilmem kaç byte'lık bir ekran silme rutini yazmaktan kurtaracaktır.

Aşağıda bu adresi kullanmadan yapılan bir ekran silme rutini bulunmaktadır. C64'ün ekran belleği 1 KB olup 1024 ($0400) adresinden başlar ve 1000 byte uzunluğundadır. Fakat bilgisayar dilinde 1 KB 1024 byte demektir. Dolayısıyla elimizde fazladan bir 24 byte vardır ve bu 24 byte'ın son 8 byte'ı sprite pointer'i olarak kullanılır. Ekranı silmek demek bu adreslere ASCII kodu 32 ($20) olan "BOŞLUK" karakterini yazmak demektir. Çok basit bir şekilde bir adet LDA, 1000 adet STA ve son olarak da RTS komutunu yazarsanız ekranınız çok çabuk temizlenecektir.

```
        LDA   #$20
        STA   $0400
        STA   $0401
        :::   :::::
        STA   $07E6
        STA   $07E7
        RTS
```

Gördüğünüz gibi arka arkaya 1002 adet komut yazarak ekranı sildik. Çok kolaymış değil mi? Evet ama maalesef çok uzun. Bu arada bir not düşelim, bu şekilde yazılan programlar daha hızlı çalışır ama hafızayı da yer bitirir. Nitekim 1K'lık bir bölgeyi temizlemek için 3003 byte uzunluğunda bir program yazdık. Bence hızı bir tarafa bırakıp birazcık hafızadan tasarruf edelim ne dersiniz?

```
        LDX   #$00
        LDA   #$20
TEKRAR  STA   $0400,X
        STA   $0500,X
        STA   $0600,X
        STA   $0700,X
        INX
        BNE   TEKRAR
        RTS
```

İşte birazcık tasarruf yaptık. Sadece 3003 byte'lık ekran silme rutini birdenbire 20 byte'a düştü. Şimdi bu rutin çok mu yavaş çalışacak? Hayır. Aslında aradaki hızı anlamak imkansızdır. Çünkü zaten makine dili komutlarının yerine getirilmesi için birkaç mikrosaniye yeterlidir. Yani siz C64'te 2 mikrosaniye süren bir komuttan 500.000 tanesini arka arkaya çalıştırırsanız sadece 1 saniye zaman harcamış olursunuz. Dolayısıyla yukarıdaki iki rutinin süreleri size aynı gelecektir.

İkinci örneğimizde aslında 1000 byte değil 1024 byte temizliyoruz. 24 bayt fazla temizlik yapmanın duruma göre şöyle bir sakıncası vardır. Eğer programınızda sprite kullandıktan sonra yukarıdaki rutinle ekranı silerseniz sprite pointerleriniz bozulacaktır ve ekranda saçma sapan şekile sahip spritelar olacaktır. Bunu önlemenin yolu ise yukarıdaki rutinde INX komutundan sonra CPX #$FA komutu kullanmak ve STA komutlarındaki adresleri ayarlamaktır.

Neyse biz tasarruf etmeye devam edelim ve en iyisi yukarıdaki rutinlere hiç bulaşmadan programımızın başlangıcında bir JSR komutuyla ekran silme işini sessizce halledelim.

```
        JSR   $E544
```

İşte bu kadar. Sadece 3 bayt ile koskoca ekranı temizledik, üstelik sprite pointerlerimiz de bozulmadı. Peki bu nasıl oluyor? C64 aslında ekranı arka arkaya 1000 byte şeklinde temizlemiyor. Ekran silme işi satır satır yapılıyor. Eğer $E544'deki rutini incelerseniz $E55E'den itibaren aşağıdaki komutları görebilirsiniz.

```
E55E  LDX   #$18
E560  JSR   $E9FF
E563  DEX
E564  BPL   $E560
....  ...   .....
```

İşte asıl ekran silme işi burada yapılıyor. O zaman niçin $E55E yerine $E544 adresini kullanıyoruz? İşin aslı şu: Eğer siz programınızda $D018 adresini kullanarak ekran belleğinin yerini değiştirdiyseniz - Ekran belleği 1K'lık adımlarla o anki VIC BANK içinde 16 farklı yerde olabilir- ve bu yeni ekran adresini 256 böldükten sonra bulduğunuz değeri $0288 adresine yazarak değişikliği KERNAL'in ekran editörüne bildirmeniz gerekmektedir. Eğer BANK değiştirmişseniz bank başlangıç adresini de eklemeniz gerekmektedir. $E544 adresi işte bu değişikliklere göre ekran adresini hesaplayıp silme işini yapacaktır. Alışkanlık olduğu için her zaman $E544 kullanılır. Aslında ekran silme işini $E536 adresini kullanarak da yapabilirsiniz. Fakat önce bir LDA komutu kullanarak ekran silindikten sonra ekrana yazılacak karakterlerin rengini değiştirme işlemini de yapmış olacaksınız.

Eğer özetleyecek olursak elimizde ekran silmek için 3 adres bulunmaktadır.

1 - $E536: Bu adrese gitmeden önce bir LDA komutuyla yazı rengini değiştirerek ekran silinir.

```
        LDA   #$01
        JSR   $E536
        ...   .....
```

Burada LDA komutuyla, ekran silindikten sonra yazılar beyaz olarak ekrana yazılır. Bu işlemi eğer BASIC ile yapmak istersseniz aşağıdaki komutları kullanmanız gerekecektir.

```
POKE 780,RENKKODU : SYS 58678
```

2 - $E544: Eğer bu adresi kullanırsanız o anki yazı rengi korunur ve ekran silindikten sonraki yazılacak yazıların rengi değişmez. Bu adresin BASIC'teki karşılığı ise SYS 58692'dir.
3 - $E55E: Eğer yazdığınız programda ekran belleğinin yerini ve/veya bank değiştirmediyseniz ve yazı rengi sizin için önemli değilse bu adresi güvenle kullanabilirsiniz. Bu adres daha önce bahsettiğimiz satır satır ekran silme işini yapan adrestir. BASIC'ten SYS 58718 komutuyla aynı işi yaparsınız.

Sırada ikinci ROM adresimiz var.

ADRES: $E566
GÖREVİ: KURSÖRÜ EKRANIN SOL ÜST KÖŞESİNE YERLEŞTİRİR
KLAVYE KARŞILIĞI: CLR/HOME

Fazla teknik bilgiye gerek yok. Ekranı silmeden kursör sol üst köşeye gider. Aslında $E55E adresindeki rutinin devamında yer alır. Daha önceki verdiğimiz adreslerde ekran silindikten sonra kursör sol üst köşeye gitmektedir. Bu adresi de BASIC'ten SYS 58726 komutuyla kullanabilirsiniz.

ADRES: $E9FF
GÖREVİ: EKRANDAKİ BİR YAZI SATIRINI SİLER

İşte geldik güzel bir adrese... Bu adres gerçekten güzeldir çünkü X registerini kullanarak istediğiniz bir ekran satırını silebilirsiniz. Fakat bir şeye dikkat etmeniz gerekmektedir. Bu rutinde ilk ekran satırı 1'den değil 0'dan başlamaktadır. Dolayısıyla sizin ekranda 1,2,3 diye saydığınız satırlar artık 0,1,2 olacaktır. Bu arada ufak bir hatırlatma yapayım. Derginin bu sayısında başladığımız C64'e yeni komutlar ekleme konusundaki CLS komutu, komuttan sonraki parametrenin olup olmamasına göre $E544 veya $E9FF adresini kullanmaktadır.

Daha önce ekranın aslında satır satır silindiğinden bahsetmiştim. Bu silme işi yukarıdan aşağıya doğru değil, aşağıdan yukarıya doğrudur. Bu küçük bilgiden sonra bir iki örnek verelim.

```
        LDX   #$0C
        JSR   $E9FF
        ...   .....
```

İlk örneğimizde ekranın tam ortasındaki yazı satırını silmiş olduk.

```
        LDX   #$14
CLEAR   JSR   $E9FF
        DEX
        CPX   #$08
        BNE   CLEAR
        ...   .....
```

Bu örnekte ise ekranda 8 ile 20. satırlar arasını sildik. Ve son örnek olarak...

```
        LDX   #$10
        JSR   $E9FF
        DEX
        BPL   CLEAR
        ...   .....
```

Son örnekte ise ekranın üstten 17 satırını siliyoruz. Eğer en üstten itibaren olacak şekilde birkaç satır silecekseniz o zaman sadece aşağıdaki komutlar işinizi görecektir.

```
        LDX   SATIRNO
        JSR   $E560
        ...   .....
```

Yukarıdaki adresleri BASIC'te şu şekilde kullanabilirsiniz:
1 - Sadece bir satır silmek için:

```
POKE 781,SATIRNO : SYS 59903
```

2 - En üstten itibaren birkaç satır silmek için:

```
POKE 781,SATIRNO : SYS 58720
```

ADRES: $EEB3
GÖREVİ: 1 MİLİSANİYE GECİKME RUTİNİ

Bu adreste 1 milisaniye gecikme rutini bulunmaktadır. Eğer program içinde kullanmak isterseniz X veya Y registerini kullanmalısınız. Registere girdiğiniz sayı kadar milisaniye gecikme sağlarsınız. En fazla gecikme 0 değeri ile sağlanır ve 256 milisaniyedir. Daha uzun süreli bir gecikme elde etmek isterseniz iç içe döngü kullanmanız gereklidir. 7D3 demo party'de 256 byte on the fly olarak yaptığım "ZOOMER" programında bu adresi kullanarak ½ saniyelik bir gecikme kullandım. Aşağıdaki örnekte yaklaşık olarak 1 saniyelik gecikme elde edebilirsiniz. Eğer Y registeri de 0 olursa gecikme yaklaşık 65,5 saniye olur.

```
        LDY   #$04
        LDX   #$00
DELAY   JSR   $EEB3
        DEX
        BNE   DELAY
        DEY
        BNE   DELAY
        ...   .....
```

ADRES: $AB1E
GÖREVİ: HAFIZADAKİ BİR YAZIYI EKRANA YAZAR

Bu adresimiz ise hafızada ASCII kodlarıyla bulunan bir yazıyı ekrana basar. En fazla 255 byte uzunluğundaki bir yazıyı ekrana basabilirsiniz. Bu rutini kullanırken dikkat etmeniz gereken üç konu var. Birincisi yazınız 255 byte'ı geçmemelidir, ikincisi C64'ün yazının bittiğini anlaması için son byte'ın değeri 0 olmalıdır, sonuncusu ise hafızadaki yazının bulunduğu adresin LOW BYTE'ı Akü'ye, HIGH BYTE'ı ise Y registerine yerleştirilmelidir. İsterseniz yazınızın rengini, kursör pozisyonunu vs. bile değiştirebilirsiniz.

```
            LDA   #<YAZI
            LDY   #>YAZI
            JSR   $AB1E
            ...   .....

YAZI        .TEXT "C64 TURKIYE SAYI 4"
            .BYTE 0
```

İlk örneğimizde o anki kursör pozisyonundan itibaren "C64 TURKIYE SAYI 4" yazısı yazılacaktır.

```
            LDA   #<YAZI
            LDY   #>YAZI
            JSR   $AB1E
            ...   .....

YAZI        .BYTE 147
            .TEXT "C64 TURKIYE SAYI 4"
            .BYTE 0
```

Bu örnekte ise yine aynı yazıyı yazdırıyoruz fakat daha önceden ekran siliniyor ve yazı en üstten itibaren yazılıyor. "147" sayısı tahmin ettiğiniz gibi ekranı silmeye yarıyor.

ADRES: $A437
GÖREVİ: HATA MESAJINI YAZDIRIR

Bu adresi kullanacağınızı pek sanmıyorum. Çünkü C64 bu adresi ekrana herhangi bir hata mesajı yazdırmak için kullanır. Belki programınızda şaka amaçlı olarak kullanabilirsiniz. Yapmanız gereken X registerine 1 ila 30 arasında (30 hariç) bir sayı yükleyip bu adrese sıçramak olacaktır.

Aşağıdaki örneği kullanarak ekrana "DEVICE NOT PRESENT ERROR" mesajı çıkarabilirsiniz. Bu arada ufak bir not düşelim: Program içinde eğer bu adresi kullanırsanız, ister JSR ister JMP ile $A437'ye gidin, C64 her zaman ekrana "READY" yazarak programdan çıkacaktır. Çünkü, gerçekte hiçbir hata olmamasına rağmen, c64 hata olduğunu zannedip programın çalışmasını durduracaktır.

```
            LDX   #$05
            JMP   $A437
            ...   .....
```

ADRES: $FCE2
GÖREVİ: C64 İLK AÇILIŞTAKİ HALİNE DÖNER

Fazla söze gerek yok. Sadece basit bir resetleme işlemi :) Bakınız : SYS 64738

Bir sonraki sayıda buluşabilmek dileğiyle şimdilik bu kadar.

KARAKTER SETİ

Belki farketmişsinizdir, oyunlarda ve bazı programlarda yazılar değişik bir şekildedir ve C64'ün alışık olduğunuz karakterlerinden çok farklıdır. Peki bu nasıl oluyor ? Cevabı ise karakter seti değiştiriliyor olacaktır. Cevap bu kadar basittir ama o karakter setini hazırlamak biraz emek ister. En basitinden değişik bir "1x1" karakter seti hazırlamak için önce bir yerlerden "FONT EDİTÖR" olarak bilinen utility'lerden bulacaksınız ve elinize bir joystick alıp her karakteri baştan çizeceksiniz. Eğer eliniz (benim gibi :)) grafiğe yatkın değilse 4096 byte'ı yeniden düzenlemek tam bir işkence olacaktır. Eğer sadece rakamları, harfleri ve noktalama işaretlerini yeniden çizecekseniz işiniz biraz daha kısa sürecektir. Bu arada C64'te karakterler hafızada nerede ve ne şekilde duruyor kısa bir bilgi verelim.

C64'ün karakter seti $D000-$DFFF adresleri arasında bulunan KARAKTER ROM'da saklanmaktadır. Derginin önceki sayılarında verdiğimiz "BELLEK HARİTASI" köşesinde aynı adreslerde "I/O CHİPLERİ" ve 4K'lık bir RAM vardır. C64 mühendisleri öyle bir sistem tasarlamışlarki C64'ün ilk açılıştaki konfigürasyonunda CPU $D000-$DFFF adreslerindeki "I/O" çiplerine erişebilirken, "karakter rom"a sadece, C64'ün grafik çipi VIC-II erişebilmektedir. Burada bir çelişki var gibi gözükmektedir. Şöyleki, VIC-II'nin kendisi $D000-$D3FF adresleri arasında, Karakter Rom'u da $D000-$DFFF adresleri arasındadır. Yani bir tür adres çakışması vardır. Fakat bizim mühendisler bir hile yaparak karakter setini VIC-II'ye $1000-$1FFF adresleri arasındaymış gibi göstermişlerdir. Oysa gerçekte bu adresler tamamen kullanıcıya kalmıştır. Yani siz $1000-$1FFF arasını kullanarak karakter setini değiştiremezsiniz. VIC-II çipi 16K'lık bir belleğe erişebilir ve bu 16K'lık bölgelere BANK denir. C64'ün belleği 64K olduğu için toplam 4 adet BANK'ımız vardır ve BANK 0,BANK 1, BANK 2 ve BANK 3 olarak adlandırılır. C64 ilk açılışta $0000-$3FFF (BANK 0) arasını kullanır. Karakter Rom'u normalde $C000-$FFFF adresleri arasındaki BANK 3'ün adreslerinin bir kısmını ($D000-$DFFF) kullanmaktadır. İşte bu $D000-$DFFF adresleri BANK 0'daki $1000-$1FFF adreslerine karşılık gelmektedir ve ROM GÖRÜNTÜSÜ olarak adlandırılmaktadır. Şimdi gelelim karakterlerin hafızada ne şekilde saklandıklarına.

Normal bir karakter seti 256 karakterden oluşur ve her karakter 8 byte'lık yer kaplar. Böylece 256*8=2048 byte bir karakter setinin kaplayacağı alandır. Fakat Karakter Rom'u 4096 Byte'tır. Yani C64'ün aslında 2 tane karakter seti vardır. C64 açılışta $D000-$D7FF adresler arasındaki ilk seti kullanır. İkinci set ise $D800-$DFFF adresleri arasındadır ve setler arasındaki geçiş "SHIFT" ve "C=" (Commodore) tuşlarıyla yapılır. Hangi karakterler hangi adreslerde olduğunu öğrenmek istiyorsanız tablo-1 işinize yarayacaktır.

Sırada ise yeni oluşturduğumuz karakter setini C64'e nasıl bildireceğiz sorusunun cevabı var. Bir karakter setinin 256 karakterden oluştuğunu ve 2048 byte (2K) yer kapladığını biliyoruz. Bu durumda bir karakter seti BANK içinde 8 farklı adresten başlayabilir. İster bütün karakterleri, isterseniz sadece bir karakteri değiştirin karakter setiniz mutlaka 8 adresin birinden başlamak zorundadır. Yani kafanıza göre adres kullanamazsınız. Karakter setinin yerini $D018 (53272) adresindeki VIC-II kontrol registerinin 3 bit'i belirler. Bit 3, 2 ve 1 2K'lık bloklar halinde, karakter setinin nerede olduğunu belirler. Bit 0 dikkate alınmaz. Tablo-2'de karakter setinin yerleşebileceği adresleri bulabilirsiniz.

Tablo-2, BANK 0 için kullanılabilecek adresleri göstermektedir. Her ne kadar 8 adet yerimiz var gibi gözüksede $0000 - $07FF, $1000 - $17FF ve $1800 - $1FFF adreslerini kullanamıyoruz. $0000 - $07FF arası bölgenin $0000 - $03FF arasındaki kısmında işletim sistemi tarafından kullanılan sistem değişkenleri, mikro işlemci tarafından kullanılan STACK (Yığın) ve vektör adresleri vs. bulunmaktadır. $0400 - $07FF arası ise ekran belleği olarak kullanılmaktadır. Dolayısıyla $0000 - $07FF arasını, eğer bana işletim sistemi lazım değil, ekran belleğinide başka bir yere taşırım derseniz belki kullanabilirsiniz. Belki diyorum çünkü özellikle $0100 - $01FF arasındaki STACK denilen bölgenin içeriğini değiştirdiğiniz anda programınızın çalışması bozulabilir, C64 kilitlenebilir. Nedeni ise -hatta kendi işletim sisteminizi yazmış bile olsanız- STACK denilen bölgenin mikro işlemci tarafından mutlaka kullanılan bir bölge olmasıdır. Kısaca tablodaki ilk adresleri unutun.

Diğer adreslerden $1000 - $17FF ve $1800 - $1FFF arası ise daha önce bahsetmiş olduğumuz ROM GÖRÜNTÜSÜ adresleri olup bu adreslere yeni karakter setinizi yerleştirip $D018'e uygun değeri verseniz bile C64'ün kendi karakterlerini görürsünüz. Daha farklı bir ifadeyle **POKE 53272,20** komutu ile C64'ün normal karakter setini seçmiş olursunuz. Örnek karakter setimizle ilgili programı incelerseniz $D018'e $1C değerini verdiğimizi görebilirsiniz.

Karakterlerin Rom'daki sıralanışı karakterlerin ekran koduna göredir. Örnek verecek olursak "A" karakterinin ekran kodu "1" olup Romda $D008 - $D00F adresleri arasındadır. Bir karakter 8 byte yer kaplamaktadır. Bu durumda bir karakterin Rom'daki adresi şöyle bulunur.

```
ADRES = $D000 + KARAKTERİN EKRAN KODU * 8
```

Eğer karakter setiniz Ram'de ise formüldeki $D000 yerine karakter setinin Ramdaki başlangıç adresini yazmanız yeterli olacaktır.

Bir byte'ın toplam değeri dolu olan bitlerin "bit değeri" toplanarak bulunur. Bir karakter 8 byte olduğuna göre bütün byte'lar hesaplanarak karakterin hafıza adresine POKE komutuyla yazılır. Eğer sadece harf, rakam ve noktalama işaretlerini değiştirecekseniz 2K yerine ilk 64 karakteri değiştirmeniz

yeterlidir. Son olarak bizim programımızda bütün karakter seti "sağa italik" olarak değiştiriliyor.

```
;-----------------------------------------;
;PROGRAM ADI : C64 İÇİN YENİ KARAKTER SETİ #01
;(SAĞA İTALİK FONTLAR)                    ;
;PROGRAM TARİHİ : 24-09-2003              ;
;PROGRAMI YAZAN : İSMAİL "HADES" ŞAHİN    ;
;C64 TÜRKİYE SAYI 4 İÇİN HAZIRLANMIŞTIR   ;
;(wordpad, c64asm v1.1 ve VICE v1.11 kul-
lanılmıştır)                              ;
;-----------------------------------------;

                    *=$0801

                    .word  nextline
                    .word  2003   ; 2003
                    .byte  $9e    ; SYS
                    .text  "2061" ; 2061
                    .byte  0      ; komutu
nextline            .word  0

;-----------------------------------------;
;ROMDAKİ KARAKTER SETİ RAM'E TRANSFER EDİLECEK
;=========================================
;Transfere başlamadan önce hafıza
;düzenlenecek. Bunun için önce interruplar
;engelleniyor ve daha sonra karakter rom'unun
;CPU (6510)  tarafından okunabilmesi için I/O
;bölgesi düzenleniyor.
;-----------------------------------------;
                    sei
;İnterruptları engelle
                    lda    $01
;Karakter ROM'u
                    and    #$fb
;artık CPU tarafından
                    sta    $01
;okunabilir
;-----------------------------------------;
;     Transfer için adresler ayarlanıyor...  ;
;-----------------------------------------;
                    lda    #$d0
;ROM adresi = $D000
                    ldx    #$30
;RAM adresi = $3000
                    ldy    #$00
;olacak şekilde ayarla.
                    sty    $fb
;$FB ve $FC adreslerinde
                    sta    $fc
;ROM adresi tutuluyor.
                    sty    $fd
;$FD ve $FE adreslerinde
                    stx    $fe
;RAM adresi tutuluyor.
;-----------------------------------------;
;4096 Byte $D000 adresinden $3000 adresine
;taşınıyor.Karakter seti 4096 byte
;uzunluğundadır ve bu bilgi 4096/256 = 16
;döngüde transfer edilebilir.
;-----------------------------------------;
                    ldx    #$10
;Blok sayacı = $10 (16)
transfer            lda    ($fb),y
;ROM'dan o anki adresi
                    sta    ($fd),y
;oku ve RAM'e kopyala
                    iny
;byte sayacını 1 arttır
                    bne    transfer
;"0" değilse geri dön
                    inc    $fc
;ROM HIGH adresi 1 arttır
                    inc    $fe
;RAM HIGH adresi 1 arttır
                    dex
;Blok sayacını 1 azalt
                    bne    transfer
;"0" olmadıysa geri dön.
                    lda    $01
;I/O bölgesi eski haline
                    ora    #$04
;getiriliyor. CPU artık
                    sta    $01
;karakter roma erişemez
                    cli
;interruptlara izin ver
                    lda    #$1c
;VIC'e yeni karakter
                    sta    $d018
;setinin yerini bildir.
;-----------------------------------------;
;     Karakter setimiz sağa italik olacak    ;
;-----------------------------------------;
                    lda    #$30
;Karakter seti başlangıç
                    ldy    #$00
;adresi $3000 olarak
                    sty    $fb
;$FB ve $FC adreslerine
                    sta    $fc
;kaydediliyor
;-----------------------------------------;
;Y registeri karakterin byte sayıcısı olacak.
;Bir karakterin datası 8 byte'tan oluştuğu
;için Y registeri 8 kez arttırılıyor ve "0"
;olup olmadığı kontrol ediliyor. "0" olunca
;bir döngüde 256/8 = 32 karakter değiştirilmiş
;oluyor. Bu işlemler Blok sayıcısı kadar
;tekrarlanınca       bütün karakter seti
;değiştirilmiş oluyor.
;-----------------------------------------;
;  Değişikliğin yapıldığı asıl bölüm burası ;
;-----------------------------------------;

new_fonts           ldx    #$10
;Blok sayıcı = $10 (16)
nextchar            lda    ($fb),y
;Karakterin ilk datasını
byte1               lsr
;okuyup ve bir kez sağa
```

```
                    sta    ($fb),y
;kaydırıp geri yazıyoruz
                    iny
;2. data için arttır.
                    lda    ($fb),y
;2. datayı oku ve
byte2               lsr
;bir kez sağa kaydır
                    sta    ($fb),y
;Sonra geri yaz
                    iny
;3. data için arttır.
                    iny
;4. data için arttır.
                    iny
;5. data için arttır.
                    iny
;6. data için arttır.
                    lda    ($fb),y
;6. datayı oku ve
byte6               asl
;bir kez sola kaydır.
                    sta    ($fb),y
;Sonra geri yaz.
                    iny
;7. dta için arttır.
                    lda    ($fb),y
;7. datayı oku ve
byte7               asl
;bir kez sola kaydır.
                    sta    ($fb),y
;Sonra geri yaz.
                    iny
;8. data için arttır.
                    lda    ($fb),y
;8. datayı oku ve
byte8               asl
;bir kez sola kaydır.
                    sta    ($fb),y
;Sonra geri yaz
                    iny
;Bir sonraki karakterin
                    bne    nextchar
;ilk datası için arttır.
;"0" olmadıysa tekrar
                    inc    $fc
;RAM HIGH adresi arttır.
                    dex
;Blok sayıcısını azalt.
                    bne    nextchar
;"0" olmadıysa tekrar
                    rts
;Programı bitir.
;-----------------------------------------;
;NOT : Program listesindeki "asl" komutları
;"lsr" ve "lsr" komutları "asl" olarak
;değiştirilirse yeni karakter seti sola italik
;olur.
;-----------------------------------------;
                    .end
```

Derginin bir sonraki sayısında bütün karakter setini başaşağı çevireceğiz. İsteyen uğraşabilir ve çözümünü .txt dosyası olarak bana gönderebilir. Veya sadece program ile değişik bir karakter seti yaparsanız dergide yayınlayabilirim. Şimdiden iyi çalışmalar....

SET	ADRES	VIC-II ROM GÖRÜNTÜSÜ	AÇIKLAMA
1	D000-D1FF	1000-11FF	Büyük harf karakterler
	D200-D3FF	1200-13FF	Grafik karakterleri
	D400-D5FF	1400-15FF	Büyük harf karakterler (negatif)
	D600-D7FF	1600-17FF	Grafik karakterleri (negatif)
2	D800-D9FF	1800-19FF	Küçük harf karakterler
	DA00-DBFF	1A00-1BFF	Büyük harf ve grafik karakterler
	DC00-DDFF	1C00-1DFF	Küçük harf karakterler (negatif)
	DE00-DFFF	1E00-1FFF	Büyük harf ve grafik karakterler (negatif)

(Tablo-1)

KARAKTER SETİ ADRESİ	ONLU	$D018 DEĞERİ
$0000-$07FF	0	$10 (16)
$0800-$0FFF	2048	$12 (18)
$1000-$17FF	4096	$14 (20)
$1800-$1FFF	6144	$16 (22)
$2000-$27FF	8192	$18 (24)
$2800-$2FFF	10240	$1A (26)
$3000-$37FF	12288	$1C (28)
$3800-$3FFF	14336	$1E (30)

(Tablo-2)

Demo Music Creator 4.0

Yeni bir sayı ve yeni bir konuyla yine karşınızdayız. Bu sayıdan itibaren, Commodore 64 ile müzik yapmak isteyenler için faydalı olacağına inandığımız birtakım dökümanlar ve programların tanıtımlarına başlıyoruz. İlk olarak Commodore'umuzun içinde bulunan o muhteşem SID çipini biraz tanıyalım. Daha sonra da en çok kullanılan müzik editörlerinden biri olan DMC'nin 4.0 versiyonunu inceleyeceğiz. İşte başlıyoruz...

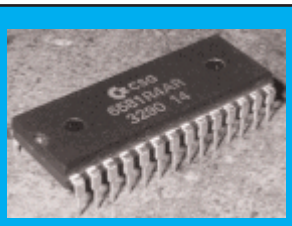
Nedir bu DMC?

DMC, 'Graffity' adlı grubun üyesi 'Brian' tarafından hazırlanan, Commodore 64 için yazılmış en güçlü müzik editörlerinden biridir. İçerisinde kendi müziğinizi yaparken kullanabileceğiniz birçok seçenek ve yardımcı editörler bulunmaktadır. DMC4.0'ı şu adreslerden indirebilirsiniz: *http://www.lemon64.com/apps/list.php?Genre=sound* veya *ftp://ftp.funet.fi/pub/cbm/c64/audio/editors/*. Kısaca birkaç özelliğinden bahsedip arkasından editörümüzü tanımaya başlayalım.

- Yapacağınız müzik dosyasının içerisinde 8 farklı parçayı saklamanız mümkün. Bu özellik genelde oyunlara müzik ve ses efektleri hazırlayanlar tarafından kullanılmaktadır. Mesela, ilk parça oyunun giriş müziği, ikincisi skor tablosu, üçüncüsü bölüm atlama kısmı, dört ve daha sonra ise oyundaki ses efektleri için kullanılabilir. DMC 4.0'da sadece 8 taneye kadar izin var. Fakat daha fazlasını kullanmanızı sağlayan editörler de mevcut.
- Synthesizer özelliği ile kullanıcı klavyesini bir org gibi kullanabilir. Bu sayede, başka bir enstrüman kullanmadan aklınızdaki bir melodiyi sadece Commodore'unuz ile çıkarma şansınız var. Alt yapınızı hazırladınız ama sololarınızda sorun mu var? İşte tam burda 'Synthesizer' yardımınıza yetişir ve 'Record' özelliği ile çaldığınız şeyleri kaydetmenize izin verir. Tek dikkat edilmesi gereken şey ise çaldığınız şeylerin 00 numaralı Sector'e kaydedildiği. Peki Sector nedir diye soranlar var sanırım. Bu kadar kısa bilgiden sonra artık kullanılan terimleri açıklamaya ve editörümüzü tanımaya başlayabiliriz.

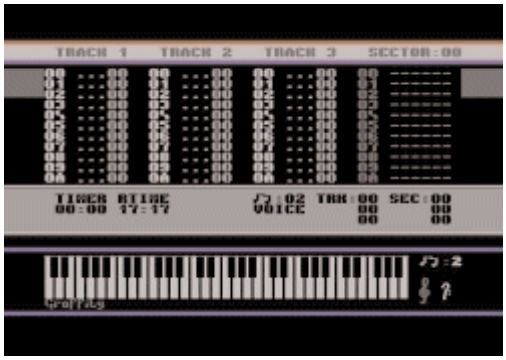
Kısa kısa...

- SECTOR: İşte burası notalarımızı yazdığımız yer. Yapacağınız müziğe ait herşeyi (baslar, arpejler, ritmler, vs.) bu sectorlere yazmanız gerekiyor. Her bir sectorün kendisine ait 256 satırlık bir alanı var. Notalarınız ve özel komutlarınız bu satırlarda yer alacak. Kullanabileceğiniz sector numaraları ise 00-3F arası ile sınırlı. Yani 64 adet. Sakın bu sayılar size az gelmesin. İşin içine girmeye başlayınca bunun çeyreğini bile kullanmadan çok güzel şeyler yapacağınızı farkedeceksiniz.
- TRACK: Kısaca 'Kanal' diyebiliriz. DMC'de kullanabileceğiniz 3 adet track mevcut. Her bir track üzerinde ise hazırladığınız sectorleri ve özel komutları yerleştirebileceğiniz 00-FF arası 256 satır bulunmakta. Track ve sector açıklamalarında bahsedilen 'Özel Komutlar'ın listesini yazının ilerleyen kısımlarında bulabilirsiniz.
- TIMER: Çalan süreyi gösterir.
- RTIME: Parçanın ne kadar Raster Time harcadığını XX:YY şeklinde gösterir. (XX o anki, YY ise maksimum raster time)
- TRK ve SEC: O an çalınan Track ve Sector bilgilerini gösterir.
- Ekranın alt tarafında gördüğünüz klavye ise eğer parçayı dinliyorsanız çalan notaları, Synthesizer ya da Record moduna ise sizin çaldığınız notaları gösterir.
- Klavyenin yanında ise oktav bilgilerini görebilirsiniz.



SID (Sound Interface Device)

1981 yılının Ocak ayında, Commodore firması, 'Project C64' adıyla yeni bir çalışma başlatmıştı. Bu çalışma sonucunda ortaya çıkarmak istedikleri şey ise tamamiyle yeni ve muhteşem bir oyun bilgisayarıydı. Fakat bu fikir, projenin gelişmesi sürecinde değişti ve Commodore firması, bu bilgisayarın gerçek bir ev bilgisayarı olmasına karar verdi. Düşüncelerine göre iyi bir ev-oyun bilgisayarı kaliteli sesler üretmeliydi. SID çipi üzerindeki çalışmalar 1981 yılının baharında Robert "Bob" Yannes ve ekibi tarafından başlatıldı. Bu arada, Robert Yannes'in VIC-20'nin dizaynına da büyük katkısı olduğunu unutmadan ekleyelim. Daha önce Synthesizer'larla ilgili konularla uğraşmış olan Yannes'in aklındaki fikir, bu synthesizer'lara benzer bir ses çipi üretmekti. Kendisi, 2 teknisyen ve bir de CAD operatöründen oluşan takımıyla yaptığı 4-5 aylık bir çalışma sonucunda muhteşem SID (6581) çipini tamamladılar. Maliyeti daha da ucuzlatabilmek için bu çip üzerindeki bazı özelliklerin çıkartılmasından sonra ise çip son haline geldi. SID çipinin gerçek kapasitesini bugün bile kimse (buna yapanlar da dahil!) tam olarak bilememektedir. Bunun en büyük sebebi ise çipte bulunan bazı hatalardan kaynaklanmaktadır. Yapan ekip bu hataları farketmiş fakat düzeltmek için yeterli zamanlarının olmamasından ötürü çip bu şekilde üretilmeye ve kullanılmaya başlanmıştır.



Editörümüzün genel görünümü

Ana Menü:

Ana Menü'ye geçmek için sadece '‹-' tuşuna basmanız yeterli olacaktır. Buradaki seçeneklerin açıklamalarına gelince;
- Sound Editor: Kullanacağınız enstrümanlara ait ayarlar.
- Filter Editor: Enstrümanlarda kullanacağınız filtrelere ait ayarlar.
- Music Setup: Volume ve Hız ayarları.
- Disk Menu: Directory listeleme, parça yükleme, saklama ve sürücünüze komut verme işlemleri için kullanacağınız yer.
- Player: Player seçmek için kullanılır. 4.0 A ve 4.0 B olarak iki tane mevcut;

Sound Editor' ün Kullanımı:

Programı ilk çalıştırdığınızda içerisinde hiçbir ses olmadığını göreceksiniz. Kendinize güveniyorsanız, enstrümanlarınızı kendiniz hazırlamaya başlayabilir ya da DMC ile yapılmış başka parçalardaki sesleri kullanmaya başlayabilirsiniz.Yapmış olduğunuz sesi test etmek için SPACE tuşuna basmanız yeterli olacaktır. İlk başlarda bol bol alıştırma yapmanız gerekecek. Hemen kısa bir örnekle devam edelim.

Bu örnekteki değerler, enstrümanın volume değeri, hızı, vibrato efektleri gibi ayarları yapmanızı sağlar. Burada ## ile gösterilen yerdeki değerler sesinizin başlayacağı adresi gösterir. SHIFT + RETURN yapalım ve ses editöründeki bu adres kısmına geçelim. Şimdi ise şu değerleri girin:

Değerleri girdikten sonra SPACE tuşuna basalım ve neler yapmışız bir duyalım. İşte yapmış olduğumuz ilk ses. Burada dikkat edilmesi gereken en önemli şey, en sona eklediğimiz 91 değeri. Bu değeri girdikten sonra sağında gri renkte 00 yazdığını göreceksiniz. Bu gördüğünüz yer, sesin çaldıktan sonra yani '41 00' değeri işlendikten sonra nereye dönüleceğini gösterir. Bütün bu değerler başlangıçta biraz karışık gelebilir ama ilerledikçe o kadarda zor olmadığını farkedeceksiniz.

Şimdi de az önce oluşturduğumuz enstrümana bir de 'FILTER' yani filtre ekleyerek bir bas enstrüman oluşturalım. '‹' tuşuna basarak ana menüye gelip 'FILTER EDITOR'e geçin ve aşağıdaki değerleri girin:

Tekrar 'SOUND EDITOR' ekranına geçelim. Şu an 00 nolu enstrümanın üzerinde olmanız lazım. Bunu ekranın sağ üst kısmındaki SND yazan yerden anlayabilirsiniz. Enstrümanlar arasında geçiş için '+' ve '-' tuşlarını kullanabilirsiniz. Şimdi '^' tuşuna basıp sesimizi hafızaya alalım. Daha sonra '+' tuşuna basarak 01 nolu sese geçin ve '@' tuşuna basın. Böylece daha önce yapmış olduğumuz 00 nolu sesi 01 numaraya kopyalamış olduk. SPACE tuşuna basarak sesinizi test edebilirsiniz. Şimdi bu kopyaladığımız sesin FX kısmına gelip 20 değerini girin ve filtreyi aktif hale getirin. 'FILTER FX' adlı yerin aktif olduğunu göreceksiniz. Bir bakın bakalım yeni bas sesimiz nasıl olmuş.

SID Teknik Özellikleri:

3 Tonlu Osilatörler
  Frekans aralığı: 0-4 kHz
Her Osilatör İçin 4 Dalga Biçimi
  Üçgen, dişli, değişken darbe, gürültü
3 Genlik Modülatörü
  Genlik aralığı: 48dB
3 Ses Üreteci
  Üstel karşılık
  Yükselme hızı: 2ms-8s
  Düşme hızı: 6ms-24s
  Durma seviyesi: 0-en yüksek volüm
  Kaybolma hızı: 6ms-24s
Osilatör Senkronizasyonu
Halka Modülasyonu
Programlanabilir Filtre
  Kesme frekansı aralığı: 30Hz-12kHz
  12dB/oktav Rolloff
  Alçak geçişli, Band geçişli, Yüksek geçişli, Notch çıkışları
  Değişken rezonans
Ana Volüm Kontrolü
2 A/D Pot Arabirimi
Rastgele Sayı/Modülasyon Üreteci
Dış Audio Girişi

Hızımızı hiç kesmeden yeni bir örnek yapalım. Şimdi '+' tuşuna basalım ve bir sonraki sesimize geçelim. Bu seferde minör bir akor yapmayalı deneyelim. Aşağıdaki değerleri girin ve yine SHIFT + RETURN yaparak ses editörüne geçin.

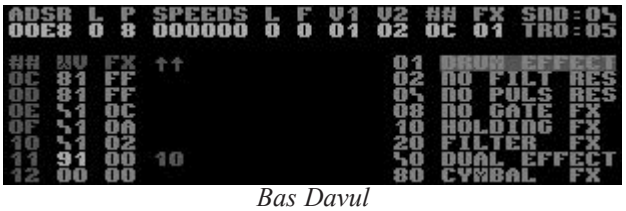
Farkettiyseniz ## 'in değeri bu sefer 02. Buraya 00 ya da 01 yazamazdık çünkü bu iki değeri bir önceki sesimizde kullandık. Şimdi kaldığımız yerden yani 02'den devam ediyoruz. Buraya da aşağıdaki değerleri girin:

Minör Akor

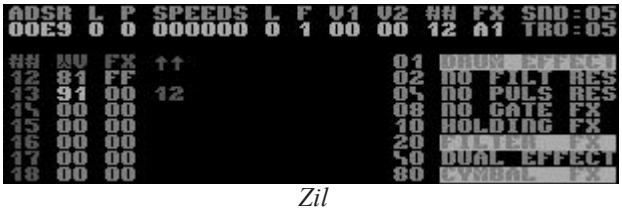
En sonda bulunan 94 değerine dikkat ederseniz bu sefer yanında 02 yazdığını göreceksiniz. Yazdığımız 4 sesi çaldıktan sonra tekrar 02 nolu adrese dönmesini bu şekilde sağlıyoruz. Peki burdaki 00-03-07-0C neyi ifade ediyor? Eğer müzikle ilginiz varsa bunun bir minör akorun açılımı olduğunu hemen anlayabilirsiniz. Major akor yapmak isteseydik 00-04-07-0C değerlerini kullanmamız yeterli olacaktı. Bu şekilde istediğiniz bütün akorları kendi parçanız içerisinde kullanabilirsiniz. Şimdi Majör akor içinde bir örnek verip diğer seslerimize geçelim. '^' tuşuyla sesimizi hafızaya alalım, '+' tuşuyla bir sonraki sese geçelim ve '@' ile 03 nolu sese bu değerleri kopyalayalım. ## kısmına 07 yazın ve aşağıdaki değerleri girin.

Majör Akor

Şimdi de davul ile ilgili seslerimizi hazırlayalım. Artık kullanacağınız tuş kombinasyonları öğrenmiş olmanız lazım. Girilecek değerlerle yazımıza devam edelim.



Bas Davul



Zil

Burda farkettiyseniz ufak bir değişiklik yaptık ve 'F ' değerine 01 girdik. Yani bu ses 1 nolu filtreyi kullanacak. 1 nolu filtrenin değerleri ise şöyle;

Kaldığımız yerden devam edip son olarak da davul sesimizi yazıyoruz.

Snare

Bu sayıdaki yazımızın sonuna geldik. Önümüzdeki sayıda DMC ile müzik hazırlarken dikkat edilmesi gereken şeyleri inceleyip biraz daha ayrıntıya gireceğiz ve hemen ardından da DMC ile yapacağımız ilk parça geliyor. Bir sonraki sayıya kadar vermiş olduğumuz örnek sesler üzerinde oynamalar yaparak kendinize ait sesler oluşturmayı deneyin ve programı baştan aşağıya kurcalayın. DMC ile yapılmış olan parçaları bulup onları da incelemeniz sizin için faydalı olacaktır. Yanlış şeyler yapmaktan korkmayın ve bizleri de yapmış olduğunuz şeylerden haberdar etmeyi unutmayın sakın. Herkese Commodore 64'lü ve bol müzikli günler...

C= + V VOLUME - Sesin şiddetini ayarlarmak için
C= + X SWITCH - Sesler arasındaki geçişleri değiştirmek için kullanılır.
^ Cursor'den sonraki sector bilgilerini hafızaya alır.
@ Hafızadaki sector bilgilerini cursor'un bulunduğu yerden itibaren yazar.
‹ Ana menü.
Shift + Return Track editor'e dönüş.
AWSEDFTGYHUJKOLP:; Notalar (C, C#, D, D#, E, F...) Buradaki nota isimlerine yabancı olabilirsiniz.ilk başta karışık gelebilir belki ama zamanla alışırsınız. Müzikle gerçek anlamda uğraşıyorsanız bunlar zaten bildiğiniz şeyler ama yeni başlayanlar için karşılıklarını verelim: C (Do), C# (Do diyez), D (Re), D# (Re diyez), E (Mi), F (Fa), F# (Fa diyez), G (Sol), G# (Sol diyez), A (La), A# (La diyez) ve B (Si).

Kısayollar (Tuş kombinasyonları):

F1 Müziği dinlemek için
F3 Çalan müziği durdurur
F4 Müziğin kaldığı yerden çalmaya devam etmesi için
F7 Müziği hızlı bir şekilde çalmak için

Track Editor:
S Yazılan özel bir komut varsa silip sectör numarası girmek için
+ Kendisinden sonra gelen sectörleri Transpoze etmek için. Yanına girilen değer kadar bütün sesler tizleşir.
- Kendisinden sonra gelen sectörleri Transpoze etmek için. Yanına girilen değer kadar bütün sesler pesleşir.
Shift+E Yaptığınız müziğin sonunu belirtir. END komutundan sonra girilen satır numarasına giderek çalmaya devam eder. Bir değişiklik yapmayıp 00 olarak bırakırsanız en baştan çalmaya devam eder. Buna kısaca LOOP diyebiliriz.
C= + E Yaptığınız müziğin sonunu belirtir. END komutundan farklı olarak müzik tamamiyle sonlanmış olur ve herhangi bir loop işlemi yapılmaz.
Shift + Return Üzerinde bulunduğunuz Sectore ait editöre geçmenizi sağlar.
F8 Synthesizer - Klavyenizi org gibi kullanmanızı sağlar.
F6 Synthesizer'da çalacağınız şeyleri kaydetmek için kullanılır. Dikkat etmeniz gereken bir şey var, kayıt edilecek bilgiler 00 nolu sector'e kayıt edilir.
‹ Ana menü.
^ Üzerinde bulunduğunuz Track bilgilerini hafızaya alır.
@ Hafızadaki track bilgilerini üzerinde bulunduğunuz track'e yazar.
Shift+C Üzerinde bulunduğunuz track bilgilerini istediğiniz bir track'e kopyalar. Yer için sizden onay bekler.
Shift+X İki track'in yerini değiştirmek için kullanılır. Yer için sizden onay bekler.
Shift+Home Üzerinde bulunduğunuz track içeriğini temizlemek için.
Shift+T Müzik seçimi (0..7)
Shift+I Müziğe ait ses, filtreler, sectorler gibi bütün bilgiler temizlenir.

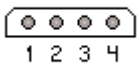
Sector Editor:
Shift + Home Sector içeriğini temizlemek için.
Shift + [Bir sonraki sector.
Shift +] Bir önceki sector.
Shift + < Cursor'den sonraki sesleri tizleştirmek için (Transpoze Up)
Shift + > Cursor'den sonraki sesleri pesleştirmek için (Transpoze Down)
+ Yazacağınız notalara ait sesleri duymak istiyorsanız bu tuşa basıp VOICE özelliğini aktif hale getirmeniz lazım.
£ GATE - Sürekli çalan bir sesi sonlandırmak için. Genelde Fade Out tipinde bir efekt verir.
- Cursor'ün üzerinde bulunduğu bilgiyi siler.
C= + D DURATION - Kendisinden sonra girilecek notaların uzunluğunu belirler.
C= + S SOUND - Kullanılan sesi seçmek için kullanılır.
C= + G GLIDE - İki ses arasında kullanılır ve 1. sesten 2.sese geçişi sağlar. Gitar çalanlar için bir nevi Bend olayı diyebiliriz.

1541-II'NİN PC PSU'SUYLA ÇALIŞTIRILMASI

Başlıktan da anlayacağınız gibi burada Commodore 1541-II (hatta 1571-II ve 1581) disket sürücülerini, bir PC power supply kullanarak nasıl çalıştırabileceğimizi anlatacağım. Ben uzun zamandır 1541-II'lerimi bu şekilde çalıştırıyorum.

Bundan 5-6 sene önce fırtınalı bir havada dışarıdaki elektrik direğindeki tellerin birbirine değmesi sonucu hem 1541-II'lerin adaptörleri hem de televizyonun besleme devresi bozuldu. Televizyondaki arızayı bir entegreyi değiştirerek giderdim. Benim asıl derdim 1541-II'ler olduğu için önce basit bir adaptör yaptım. Eğer şebeke gerilimi yeterince yüksekse sorun olmuyordu. Fakat şebeke gerilimi dalgalanmaya başladığında veya düştüğünde sürücülerde okuma problemi başlıyordu. Bunu evinizdeki odanın lambasından anlayabilirsiniz. Eğer lambanın parlaklığı durup dururken azalıp artıyorsa dalgalanma var demektir ve bu dalgalanma bazı elektrikli cihazlarda arızaya yol açabilir. Veya o gün lambanın parlaklığı az ise, anlayın ki şebeke gerilimi düşüktür. Normalde 220 Vac olması gereken değerin bazen 170 Vac'ye kadar düştüğünü bilirim. Bu nedenle aklıma PC powerinden çalıştırma fikri geldi. Bir arkadaştan aldığım eski bir PC power kartından (kart kutu içinde değildi) 1541-II için gerekli gerilimleri bulup, lehim işlerini de yaptıktan sonra bir kutuya koydum ve 1541-II'lerimi geçen seneye kadar kullandım. Daha sonra ise bit pazarından 1.5 milyona aldığım PC poweri ile kullanmaya başladım. Bu hikayeden sonra artık konuya işin teknik kısmına girebiliriz.

Harddisk'lerin çalışma gerilimi ile 1541-II sürücünün çalışma gerilimi aynı olup +5 Vdc ile +12 Vdc yeterlidir. Fakat buradaki asıl sorun bağlantının nasıl yapılacağıdır. Tamamen farklı soketler olduğu için en kolay yöntem, eğer 1541-II'nin adaptörü bozuksa, adaptörden 1541-II'ye giden kabloyu adaptör tarafından kesmek ve harddiske takılan power soketini de kestikten sonra kabloların ucunu açmak ve uygun yerlere bağlamaktır. Fakat hangi kablo nereye bağlanacak onu bilmek lazım. Aksi takdirde 1541-II'ye elveda diyebilirsiniz. Bu nedenle eğer eliniz elektrik-elektronik işlerine yatkın değilse ortaya çıkacak arızalardan dolayı sorumluluk tamamen size aittir. Neyse, bu kadar laf kalabalığı yeter diyelim ve işe başlayalım. Bir PC power supply'sinde harddiske takılan power kablosu aşağıdaki şekildedir. 45°'lik köşeler üstte olacak ve kablolar arkada kalacak şekilde soketi kendinize doğru tutarsanız bacak pin sıralanışı aşağıdaki gibi olacaktır. Tabloda gördüğünüz kablo renkleri tamamen standarttır.



PİN	İSİM	RENK	AÇIKLAMA
1	+12V	SARI	+12 VDC
2	GND	SİYAH	TOPRAK
3	GND	SİYAH	TOPRAK
4	+5V	KIRMIZI	+5 VDC

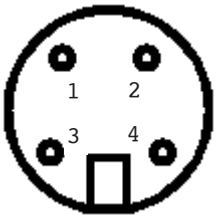
Şimdi elinizde bozuk bir 1541-II adaptörü olduğunu ve yukarıdaki gibi kesme işlemini yaptığınızı kabul edelim. Ama önce kablo renklerine bir bakalım. Bendeki kayıtlara göre 1541-II için farklı kablo renkleri olan iki soket var.

```
1-SİYAH: TOPRAK          1-SİYAH: TOPRAK
2-BEYAZ: +5 Vdc          2-YEŞİL: +5 Vdc
3-KIRMIZI: +12 Vdc       3-KIRMIZI: +12Vdc
```

Eğer elinizdeki kabloların rengi bunlardan biriyse harddisk power kabloları ile bağlantı yapmaya başlayabiliriz. Aşağıdaki tabloya göre bağlantıları güzel bir şekilde yaparsanız 1541-II'niz emrinize hazırdır. Güle güle kullanın...

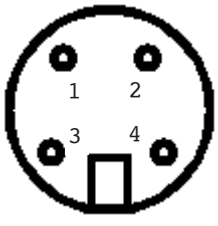
1541-II KABLOSU	HARDDİSK KABLOSU
SİYAH	SİYAH
BEYAZ/YEŞİL	KIRMIZI
KIRMIZI	SARI

Peki ya bozuk bir adaptörünüz yoksa ne yapacaksınız? Burada artık tamamen havya-lehim kullanmak zorundasınız. 1541-II'nin power girişine göre erkek soket alacaksınız -bunun için 1541-II'yi yanınızda götürmenizi tavsiye ederim- ve uygun bir kablo ile lehimleme işine başlayacaksınız. Bunun yanında bir de ölçü aleti bulabilirseniz çok iyi olur. Öyleyse hiç beklemeden pamuk eller havyaya :). Şunu söylemek istiyorum, lehim konusunda acemiyseniz bu işe hiç kalkışmayın. Veya çevrenizde bu işlerden anlayan varsa onunla konuşun. Aksi takdirde zaten zor bulunan 1541-II'yi çöpe atmak zorunda kalabilirsiniz.



1- BOŞ
2- TOPRAK
3- +12Vdc
4- +5Vdc

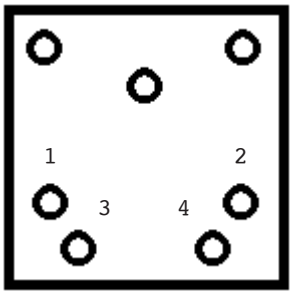
(ÖNDEN GÖRÜNÜŞ)



1- TOPRAK
2- BOŞ
3- +5Vdc
4- +12Vdc

(LEHİM TARAFINDAN GÖRÜNÜŞ)

Lehim yaparken lehim tarafından görünüş şeklini kullanın. Eğer 4'lü DIN soket bulamazsanız, 7'li soketten dönüştürmeyi deneyin. Son olarak hiçbir soket bulamıyorsanız 1541-II'yi PC poweri ile çalıştırabilmek için yapmanız gereken son bir iş kaldı. 1541-II'nin içini açarak cihazın açma kapama anahtarının veya power soketinin bacaklarına kablo lehimlemeniz gerekiyor. Bir kez daha hatırlatıyorum. Kabloları yanlış bağlarsanız 1541-II'nize cenaze töreni yapmak zorunda kalabilirsiniz.



1- +12Vdc
2- +5Vdc
3- BOŞ
4- TOPRAK

Yukarıdaki şekil 1541-II'nin power soketinin bacaklarının lehim yerlerini göstermektedir. Yani lehim tarafından görünüştür. Harddisk power kablolarını renklerine dikat ederek uygun yerlere lehimlerseniz işiniz tamam demektir. Renkleri bir kez daha yazıyorum.

KIRMIZI: +5Vdc
SARI: +12Vdc
SİYAH: Toprak

Lehim yaptığınız yerleri bir kez daha kontrol edin. Eğer %100 doğruysa 1541-II'e enerji verin. Ön paneldeki kırmızı ve yeşil ışıklar yanacak ve daha sonra yeşil ışık sönecektir. Şimdi disketi yerleştirin ve önce **LOAD "$",8** komutu sonra da **LIST** komutunu verin. Demek ki 1541-II çalışıyormuş. Güle güle kullanın.

PROGRAM DÖKÜMLERİ

Bu sayıdan itibaren dergide verdiğimiz programları, bilgisayara daha kolay girebilmeniz için CHECKSUMMER V1.0 formatında vermeye başlıyoruz. Bu listeleri bilgisayarınıza girebilmeniz için daha önce CHECKSUMMER V1.0 yüklemeniz ve çalıştırmanız gerekmektedir. Herkese kolay gelsin.

NOT: İlk 4 programın başlangıç adresleri $0801 olacak şekilde değiştirilmiştir.

SAYI 1:

PROGRAM ADI: VERTICAL BAR 1 0801 0849

```
0801: 0B 08 D3 07 9E 32 30 36  - 4391
0809: 31 00 00 00 78 A9 00 8D  - 4AB5
0811: 11 D0 A9 00 8D 20 D0 A9  - 7DB6
0819: 01 8D 20 D0 A9 02 8D 20  - 57AE
0821: D0 A9 03 8D 20 D0 A9 04  - 6886
0829: 8D 20 D0 A9 05 8D 20 D0  - 7B9E
0831: A9 06 8D 20 D0 A9 07 8D  - 6F31
0839: 20 D0 A9 08 8D 20 D0 A9  - 7F9D
0841: 09 8D 20 D0 4C 13 08 A9  - 59AE
```

PROGRAM ADI: VERTICAL BAR 2 0801 0829

```
0801: 0B 08 D3 07 9E 32 30 36  - 4391
0809: 31 00 00 00 78 A9 00 8D  - 4AB5
0811: 11 D0 A9 00 A2 07 A0 02  - 4E77
0819: 8D 20 D0 8E 20 D0 8C 20  - 6E3F
0821: D0 4C 13 08 00 00 00 00  - 1611
```

SAYI 2:

PROGRAM ADI: HEX-DEC CEVIRICI 0801 0991

```
0801: 0B 08 D3 07 9E 32 30 36  - 4391
0809: 31 00 00 00 A9 0F 8D 20  - 37D8
0811: D0 8D 21 D0 A9 0B 20 36  - 5A6E
0819: E5 A9 E1 A0 08 20 1E AB  - 73E6
0821: 18 A2 07 A0 09 20 F0 FF  - 874B
0829: A9 24 20 D2 FF A9 00 85  - 7E84
0831: CC 20 5C 08 85 FC 20 5C  - 67F3
0839: 08 85 FB A9 3D 20 D2 FF  - 9C83
0841: A5 FC A6 FB 20 CD BD 20  - 9028
0849: E4 FF C9 20 D0 F9 A2 00  - 8EDF
0851: 9D 22 05 E8 E0 0B D0 F8  - 9EDD
0859: 4C 21 08 20 71 08 AD C0  - 62A3
0861: 08 0A 0A 0A 0A 85 FB 20  - 4750
0869: 71 08 AD C0 08 05 FB 60  - 6D68
0871: 20 E4 FF C9 30 90 F9 C9  - B18E
0879: 3A B0 07 48 38 E9 30 4C  - 5B5C
0881: 8F 08 C9 41 90 EA C9 47  - 8753
0889: B0 E6 48 38 E9 37 8D C0  - 9091
0891: 08 0A AA BD C1 08 85 FD  - 8E18
0899: BD C2 08 85 FE A0 00 B1  - 8969
08A1: FD 8D BE 08 AD BF 08 91  - 807F
08A9: FD 68 20 D2 FF A0 A0 A2  - A826
08B1: 00 20 B3 EE 88 D0 F8 AD  - AB8A
08B9: BE 08 91 FD 60 00 07 00  - 43A5
08C1: 29 D8 2B D8 2D D8 2F D8  - 8D10
08C9: 79 D8 7B D8 7D D8 7F D8  - B010
08D1: C9 D8 CB D8 CD D8 CF D8  - D310
08D9: 19 D9 1B D9 1D D9 1F D9  - 86A0
08E1: 8E D5 C3 B2 C3 B2 C3 B2  - B884
08E9: C3 C9 0D C2 30 C2 31 C2  - 896C
08F1: 32 C2 33 C2 0D AB C3 DB  - 9845
08F9: C3 DB C3 DB C3 B3 0D C2  - A7CF
0901: 34 C2 35 C2 36 C2 37 C2  - 84B0
0909: 0D AB C3 DB C3 DB C3 DB  - C0DA
0911: C3 B3 0D C2 38 C2 39 C2  - 8A0A
0919: 41 C2 42 C2 0D AB C3 DB  - 9A9D
0921: C3 DB C3 DB C3 B3 0D C2  - A7CF
0929: 43 C2 44 C2 45 C2 46 C2  - 8B40
0931: 0D AB C3 B1 C3 B1 C3 B1  - AC58
0939: C3 B1 C3 C3 C3 C3 C3 C3  - C1AA
0941: C3 C3 C3 C3 C3 C9 0D C2  - A67D
0949: 20 48 45 58 20 2D 20 44  - 36D0
0951: 45 43 20 43 45 56 49 52  - 4765
0959: 49 43 49 C2 0D C2 4B 41  - 5EF6
0961: 53 49 4D 2F 32 30 30 32  - 3760
0969: 20 28 43 29 48 41 44 45  - 3CAA
0971: 53 C2 0D CA C3 C3 C3 C3  - ABEE
0979: C3 C3 C3 C3 C3 C3 C3 C3  - C300
0981: C3 C3 C3 C3 C3 C3 C3 CB  - C4D8
0989: 05 0D 00 00 00 00 00 00  - 014C
```

NOT : Yaptığım kontrolde "hex-dec" programının assembler listesinde hata olduğunu farkettim. Bu hata sonucunda çevirme işleminden sonra space'ye basılıp yeni çevirme işleminden önce eski değerlerin silinmesi sırasında ekranda başka yer silinmektedir. Yukarıdaki dökümde hata giderilmiştir.

SAYI 3:

PROGRAM ADI: SIFRE-DESIFRE 0801 0909

```
0801: 0B 08 D3 07 9E 32 30 36  - 4391
0809: 31 00 00 00 4C 13 08 4C  - 2200
0811: 6D 08 A9 00 A2 09 8D D8  - 744C
```

```
0819: 08 8E D9 08 A9 00 A2 0C  - 50A6
0821: 85 FB 86 FC A2 00 A0 00  - 714A
0829: 20 D7 08 0A 0A 8D DC 08  - 4E9A
0831: E8 20 D7 08 20 CB 08 0D  - 4B7F
0839: DC 08 91 FB 20 D7 08 20  - 6445
0841: D2 08 8D DC 08 E8 20 D7  - 8D4E
0849: 08 4A 4A 0D DC 08 C8 91  - 6B64
0851: FB 20 D7 08 20 D0 08 8D  - 6AFB
0859: DC 08 E8 20 D7 08 0D DC  - 77A6
0861: 08 C8 91 FB E8 C8 CC DB  - C4FB
0869: 08 D0 BD 60 A9 00 A2 0C  - 5D00
0871: 8D D8 08 8E D9 08 A9 00  - 60B3
0879: A2 0B 85 FB 86 FC A2 00  - 828D
0881: A0 00 20 D7 08 4A 4A 91  - 5ECA
0889: FB 20 D7 08 29 03 20 D2  - 6230
0891: 08 8D DC 08 E8 20 D7 08  - 64AA
0899: 20 CB 08 0D DC 08 C8 91  - 70A1
08A1: FB 20 D7 08 29 0F 0A 0A  - 324E
08A9: 8D DC 08 E8 20 D7 08 20  - 604C
08B1: C9 08 0D DC 08 C8 91 FB  - 96FA
08B9: 20 D7 08 29 3F C8 91 FB  - 8D0D
08C1: C8 E8 EC DB 08 D0 BB 60  - 9F9E
08C9: 4A 4A 4A 4A 4A 4A 60 0A  - 3EC6
08D1: 0A 0A 0A 0A 0A 60 BD 00  - 30CB
08D9: 00 60 03 00 00 00 00 00  - 0765
08E1: 00 00 00 00 00 00 00 00  - 0000
08E9: 00 00 00 00 00 00 00 00  - 0000
08F1: 00 00 00 00 00 00 00 00  - 0000
08F9: 00 00 00 00 00 00 00 03  - 00B1
0901: 2D 36 34 20 20 20 00 00  - 17CB
```

NOT : Şifre - Deşifre programında değişiklik yapılmıştır. Şifrelenecek yazının bulunduğu adres $0A00 yerine $0900 olarak değiştirilmiştir. Liste bu değişikliğe göredir.

SAYI 4:

PROGRAM ADI: KARAKTER SET 1 0801 0871

```
0801: 0B 08 D3 07 9E 32 30 36  - 4391
0809: 31 00 00 00 78 A5 01 29  - 333E
0811: FB 85 01 A9 D0 A2 30 A0  - 8AF0
0819: 00 84 FB 85 FC 84 FD 86  - A869
0821: FE A2 10 B1 FB 91 FD C8  - BC58
0829: D0 F9 E6 FC E6 FE CA D0  - E25B
0831: F2 A5 01 09 04 85 01 58  - 4593
0839: A9 1C 8D 18 D0 A9 30 A0  - 7ADD
0841: 00 84 FB 85 FC A2 10 B1  - 90B3
0849: FB 4A 91 FB C8 B1 FB 4A  - AAAD
0851: 91 FB C8 C8 C8 C8 B1 FB  - D034
0859: 0A 91 FB C8 B1 FB 0A 91  - 955D
0861: FB C8 B1 FB 0A 91 FB C8  - B84F
0869: D0 DD E6 FC CA D0 D8 60  - BEAB
```

PROGRAM ADI: HADES' BASIC C000 C0B8

```
C000: A9 0B A2 C0 8D 08 03 8E  - 63C8
C008: 09 03 60 A5 7A A6 7B 85  - 7541
C010: FB 86 FC A9 9D A2 C0 85  - AC3A
C018: FD 86 FE A2 00 A0 00 20  - 626B
C020: 73 00 85 02 B1 FD C5 02  - 6DD3
C028: D0 13 C8 98 DD AE C0 D0  - B502
C030: EE 8A 0A AA AD 96 C0 48  - 882D
C038: BD 95 C0 48 60 BD AE C0  - A013
C040: 18 65 FD 85 FD 90 02 E6  - 97AA
C048: FE E8 EC AD C0 D0 D5 A5  - C857
C050: FB A6 FC 85 7A 86 7B 4C  - 8A0D
C058: E4 A7 20 73 00 F0 12 20  - 5862
C060: 9E B7 E0 01 90 11 E0 1B  - 6A4E
C068: B0 0D CA 20 FF E9 4C AE  - 9741
C070: A7 20 44 E5 4C AE A7 4C  - 7C21
C078: 48 B2 20 9B B7 8E 20 D0  - 862C
C080: 4C AE A7 20 9B B7 8E 86  - 876F
C088: 02 4C AE A7 20 9B B7 8E  - 80A3
C090: 21 D0 4C AE A7 59 C0 79  - 87E4
C098: C0 82 C0 8B C0 43 4C 53  - 77A1
C0A0: 42 B0 44 45 52 49 4E 4B  - 51A1
C0A8: 50 41 50 45 52 04 03 05  - 2548
C0B0: 03 05 00 00 00 00 00 00  - 0092
```

CHECKSUMMER V1'i ve bu sayfadaki programları *http://hades6510.sitemynet.com* adresinde dosyalar bölümünden indirebilirsiniz. (c64tr_disket isimli zip dosyası)