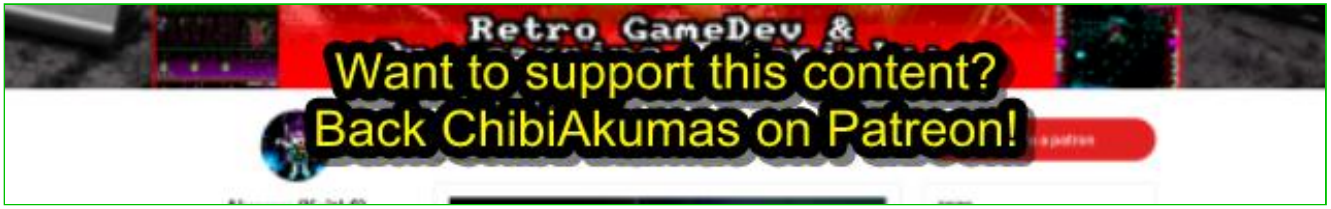

View Options
Default Dark
Simple (Hide this menu)
Print Mode (white background)

Top Menu
Main Menu
Youtube channel
Forum
AkuSprite Editor
Dec/Bin/Hex/Oct/Ascii Table

Z80 Content
Z80 Tutorial List
Learn Z80 Assembly
Hello World
Advanced Series
Multiplatform Series
Platform Specific Series
ChibiAkumas Series
Grime Z80
Z80 Downloads
Z80 Cheatsheet
Sources.7z
DevTools kit

6510 Assembly programming for the Commodore 64

The C64 is one of the most popular computers of all time, although limited to just 64k, it rivalled its competitors with hardware sprites and scrolling,

its 6510 CPU is a 6502 with built in IO ports... the is no programming difference

	C64	C128
Cpu	1mhz 6510 (6502 base)	2mhz 8502
Ram	64k	128k
Sprites	8 per line (24x21 px)	8 per line (24x21 px)
Resolution	320x200 / 160x200	320x200 / 160x200
Colors	4 per 8x8 tile from 16	4 per 8x8 tile from 16
Sound chip	SID	SID

ChibiAkumas Tutorials

	Lesson H2 - Hello World on the C64
	Lesson S2 - Bitmap Drawing on the C64
	Lesson P9 - Bitmap Functions on the C64
	Lesson P30 - Sound on the C64
	Lesson P36 - Hardware Sprites on the C64

**** Linux and Mac users! ****

Viewer "Kevin Thomas" has done a lot of work porting the C64 Chibiakumas tutorials to Linux and Unix - so if you're using those systems, you should probably check out his work over Here!

Text Graphics

The characters shown onscreen are selected by the bytes in the memory range $0400-$07FF, the colors of the tiles are selected per 8x8 square, from the registers at $D800-$DBE7... only the Low nibble of this area is used.

Bitmap Graphics

There are two modes for Bitmap graphics on the C64

Z80 Platforms

Amstrad CPC
Elan Enterprise
Gameboy & Gameboy Color
Master System & GameGear
MSX & MSX2
Sam Coupe
TI-83
ZX Spectrum
Spectrum NEXT
Camputers Lynx

6502 Content

6502 Tutorial List
Learn 6502 Assembly
Advanced Series
Platform Specific Series
Hello World Series
Grime 6502

6502 Downloads

6502 Cheatsheet
Sources.7z
DevTools kit

6502 Platforms

Apple IIe
Atari 800 and 5200
Atari Lynx
BBC Micro
Commodore 64
Commander x16
Super Nintendo (SNES)
Nintendo NES / Famicom
PC Engine (Turbografx-16)
Vic 20

68000 Content

Normal mode is 320x200... it has 2 colors per 8x8 tile, the "Bitmap data" is typically located between $2000-$3FFF, this is a 1 bpp bitmap, each tile will get its background color from the low nibble of $D020, and it:s foreground color from the low nibble of $D800-$DBE7

Bits	Detail	Address
0	Text Screen Mem - Low nibble	$0400-$07FF ----CCCC
1	Text Screen Mem - High nibble	$0400-$07FF CCCC----

Multicolor Mode is 160x200 , it has 4 colors per 4x8 tile, but setting those colors is more tricky... again it uses a bitmap screen at $2000-$3FFF, but is 2bpp... it uses a 160x200, 2 bits for each pixel choose a color from 1 of 4 locations

Bits	Detail	Address
00	Background Color	$D021
01	Text Screen Mem - Low nibble	$0400-$07FF ----CCCC
10	Text Screen Mem - High nibble	$0400-$07FF CCCC----
11	Color Memory - Low Nibble	$D800-$DBFF ----CCCC

The Border Color is defined by $D020

Graphics Memory and ports

Address	Description	Bits	Meaning
$0400-$07E7	Default area of screen memory		(1000 bytes).
$2000-$3FFF	BMP Screen Ram		
$D000-$D7FF	Char ROM in uppercase/graphics character set		(2048 bytes, 256 entries)
$D800-$DFFF	Char ROM in lowercase/uppercase character set		(2048 bytes, 256 entries)
$D011	Screen control register #1.	LXMSHVVV	L=Cur Line X=extended BG M=mode (Txt/Bmp)S=screen on H=height V=Vert scroll
$D016	Screen control register #2	---MWHHH	M=Multicolor W=scr width H=horiz scroll
$D018	Memory setup register.	SSSSTTT-	T=Text/Bmp screen address S=Screen (color) address
$D020	Border color	----CCCC	C=color
$D021	Background color	----CCCC	C=color

$D022	Extra background color #1	----CCCC	C=color
$D023	Extra background color #2	----CCCC	C=color
$D024	Extra background color #3	----CCCC	C=color
$D800-$DBE7	Color RAM	----CCCC	C=color (1000 bytes).

Palette

0	1	2	3	4	5	6	7
8	9	A	B	C	D	E	F

C64 Sprites

The Sprite pointers for the bitmap data are a single byte... multiplying the sprite pointer by 64 will give the address of the sprite *within the 16k bank of Vram* (so must be in the range $0000-$3FFF)...
$1000-$2000 and $9000-$A000 are seen by the VIC as character ROM, so sprites cannot be in this area!
We can move our screen base to something more convenient... so for example with a screen base of $4000 (Screen ram at $6000)- our sprites can be at $5000

Sprites are 21 vertical lines and 63 bytes each...
In 1bpp (2 color) mode this makes sprites 24x63...
In 2bpp (4 color) mode they are 12x63...

In both modes, Color 0 is Transparent

In 2bpp mode color 1,2 are read from $D025/6... and color 3 is the sprite color.

Address	Purpose	Bits	Meaning
$07F8-$07FF	Sprite pointers (default - will change if screen moved)	SSSSSSSS	s*64=memory address
$D000	Sprite #0 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D001	Sprite #0 Y-coordinate	YYYYYYYY	
$D002	Sprite #1 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D003	Sprite #1 Y-coordinate	YYYYYYYY	
$D004	Sprite #2 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D005	Sprite #2 Y-coordinate	YYYYYYYY	
$D006	Sprite #3 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D007	Sprite #3 Y-coordinate	YYYYYYYY	

$D008	Sprite #4 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D009	Sprite #4 Y-coordinate	YYYYYYYY	
$D00A	Sprite #5 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D00B	Sprite #5 Y-coordinate	YYYYYYYY	
$D00C	Sprite #6 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D00D	Sprite #6 Y-coordinate	YYYYYYYY	
$D00E	Sprite #7 X-coordinate	XXXXXXXX	(only bits #0-#7).
$D00F	Sprite #7 Y-coordinate	YYYYYYYY	
$D010	Sprite #0-#7 X-coordinates	76543210	(bit #8)
$D015	Sprite enable register	76543210	1=on
$D017	Sprite double height register	76543210	
$D01B	Sprite priority register	76543210	
$D01C	Sprite multicolor mode register	76543210	0=2 color 1=4color
$D01D	Sprite double width register	76543210	
$D01E	Sprite-sprite collision register	76543210	
$D01F	Sprite-background collision reg	76543210	
$D025	Sprite extra color #1	----CCCC	
$D026	Sprite extra color #2	----CCCC	
$D027	Sprite #0 color	----CCCC	
$D028	Sprite #1 color	----CCCC	
$D029	Sprite #2 color	----CCCC	
$D02A	Sprite #3 color	----CCCC	
$D02B	Sprite #4 color	----CCCC	
$D02C	Sprite #5 color	----CCCC	
$D02D	Sprite #6 color	----CCCC	
$D02E	Sprite #7 color	----CCCC	

SID sound chip

The SID chip uses memory addresses $D400-$D41C

Address	Description	Bits	Meaning
$D400	Voice #1 frequency L	LLLLLLLL	
$D401	Voice #1 frequency H	HHHHHHHH	Higher values=higher pitch

$D402	Voice #1 pulse width L	LLLLLLLL	
$D403	Voice #1 pulse width H	----HHHH	
$D404	Voice #1 control register	NPST-RSG	Noise / Pulse / Saw-tooth / Triangle / - test / Ring mod / Sync /Gate
$D405	Voice #1 Attack and Decay length	AAAADDDD	Attack / Decay (0=fastest)
$D406	Voice #1 Sustain volume and Release length.	VVVVRRRR	Sustain Volume / Release (0=fastest)
$D407	Voice #2 frequency L	LLLLLLLL	
$D408	Voice #2 frequency H	HHHHHHHH	Higher values=higher pitch
$D409	Voice #2 pulse width L	LLLLLLLL	
$D40A	Voice #2 pulse width H	----HHHH	
$D40B	Voice #2 control register	NPST-RSG	Noise / Pulse / Saw-tooth / Triangle / - test / Ring mod / Sync /Gate
$D40C	Voice #2 Attack and Decay length	AAAADDDD	Attack / Decay (0=fastest)
$D40D	Voice #2 Sustain volume and Release length.	VVVVRRRR	Sustain Volume / Release rate (0=fastest)
$D40E	Voice #3 frequency L	LLLLLLLL	
$D40F	Voice #3 frequency H	HHHHHHHH	Higher values=higher pitch
$D410	Voice #3 pulse width L	LLLLLLLL	
$D411	Voice #3 pulse width H	----HHHH	
$D412	Voice #3 control register.	NPST-RSG	Noise / Pulse / Saw-tooth / Triangle / - test / Ring mod / Sync /Gate
$D413	Voice #3 Attack and Decay length.	AAAADDDD	Attack / Decay (0=fastest)
$D414	Voice #3 Sustain volume and Release length.	VVVVRRRR	Sustain Volume / Release (0=fastest)
$D415	Filter cut off frequency L	-----LLL	Cut off frequency
$D416	Filter cut off frequency H	HHHHHHHH	Cut off frequency
$D417	Filter control	RRRREVVV	R=Resonance (0=off) / External / V= Voice 3-1
$D418	Volume and filter modes	MHBLVVVV	Mute3 / Highpass / Bandpass / Lowpass / Volume (0=silent)
$D41B	Voice #3 waveform output. (Read only)	DDDDDDDD	
$D41C	Voice #3 ADSR output. (Read only)	DDDDDDDD	

Recent New Content

Amiga - ASM PSET and POINT for Pixel Plotting

Learn 65816 Assembly: 8 and 16 bit modes on the 65816

SNES - ASM PSET and POINT for Pixel Plotting

ARM Assembly Lesson H3

Lesson P65 - Mouse reading on the Sam Coupe

Mouse Reading in MS-DOS

Risc-V Assembly Lesson 3 - Bit

ops and more maths!

Mouse reading on the MSX

Hello World on RISC-OS

Atari 800 / 5200 - ASM PSET and POINT for Pixel Plotting

Apple 2 - ASM PSET and POINT for Pixel Plotting

Making a 6502 ASM Tron game...
Photon1 - Introduction and Data Structures

Gaming + more:

Emily The Strange (DS) - Live full playthrough

$150 calculator: Unboxing the Ti-84 Plus CE (eZ80 cpu)

Recent New Content

Amiga - ASM PSET and POINT for Pixel Plotting

Learn 65816 Assembly: 8 and 16

bit modes on the 65816

SNES - ASM PSET and POINT for Pixel Plotting

ARM Assembly Lesson H3

Lesson P65 - Mouse reading on the Sam Coupe

Mouse Reading in MS-DOS

Risc-V Assembly Lesson 3 - Bit ops and more maths!

Mouse reading on the MSX

Hello World on RISC-OS

Atari 800 / 5200 - ASM PSET and POINT for Pixel Plotting

Apple 2 - ASM PSET and POINT for Pixel Plotting

Making a 6502 ASM Tron game...
Photon1 - Introduction and Data Structures

Gaming + more:

Emily The Strange (DS) - Live full playthrough

$150 calculator: Unboxing the Ti-84 Plus CE (eZ80 cpu)

ASM Tutorials for
Z80,6502,68000
8086,ARM and
more On my
Youtube Channel

Questions,
Suggestions
Advice?
Discuss on the
Forums!

Recent New Content

[Amiga - ASM PSET and POINT for Pixel Plotting]

[Learn 65816 Assembly: 8 and 16 bit modes on the 65816]

[SNES - ASM PSET and POINT for Pixel Plotting]

[ARM Assembly Lesson H3]

[Lesson P65 - Mouse reading on the Sam Coupe]

[Mouse Reading in MS-DOS]

[Risc-V Assembly Lesson 3 - Bit ops and more maths!]

[Mouse reading on the MSX]

[Hello World on RISC-OS]

[Atari 800 / 5200 - ASM PSET and

POINT for Pixel Plotting

Apple 2 - ASM PSET and POINT for Pixel Plotting

Making a 6502 ASM Tron game...
Photon1 - Introduction and Data Structures

Gaming + more:

Emily The Strange (DS) - Live full playthrough

$150 calculator: Unboxing the Ti-84 Plus CE (eZ80 cpu)