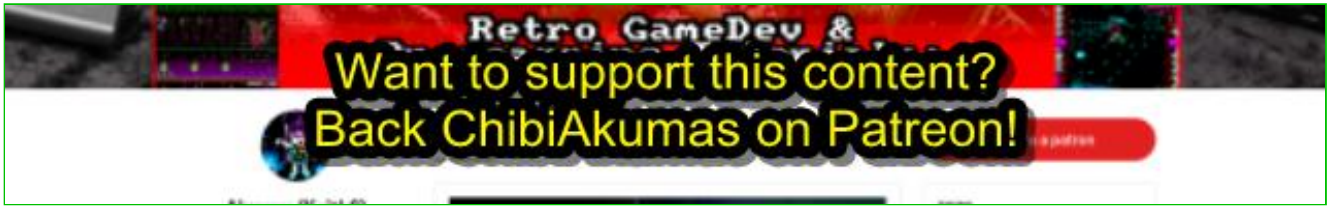

View Options
Default Dark
Simple (Hide this menu)
Print Mode (white background)

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

6502 Assembly programming for the Atari 800 & 5200

The early Atari games consoles weren't a true 6502, but by the time of the Atari 5200 things had gotten pretty good!

The Atari 800 and 5200 are basically the same machine, and we can write games that work on both pretty easilly... in fact the differences are INTENTIONAL.. they didn't want games to work on both the computer and console, as they were different operating companies, so the address of the sound and graphics chips were moved to stop the games working!

We'll be covering the Atari 5200 console, and Atari 800 home computer in these tutorials.

	Atari 5200	Atari 800
Cpu	1.79 mhz 6502	1mhz 65C02
Ram	16k	48k
Resolution	320x200 @ 2 color	320x200 @ 2 color

	160x200 @ 4 color	160x200 @ 4 color
Sprites	8x128 pixel sprites... 5 onscreen	8x128 pixel sprites... 5 onscreen
Sound	4 channel POKEY	4 channel POKEY

ChibiAkumas Tutorials

	Lesson P2 - Bitmap Functions on the Atari 800 / 5200
	Lesson P11 - Joystick Reading on the Atari 800 / 5200
	Lesson P18 - Palette definitions on the Atari 800 / 5200
	Lesson P23 - Sound on the Atari 800 / 5200

Z80 Platforms
Amstrad CPC
Elan Enterprise
Gameboy & Gameboy Color
Master System & GameGear
MSX & MSX2
Sam Coupe
TI-83
ZX Spectrum
Spectrum NEXT
Computers 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

Differences between the Atari 800 and 5200

The Atari 800 and 5200 have some differences, The GTIA (graphics) and POKEY (sound and io) are at different addreses, also the default cartridge base address and size are different.
The 800 cartridges are 8K, the 5200 has 32k ones.

	Atari 5200	Atari 800
Cart ROM	$4000	$A000
GTIA (Graphics)	$C000	$D000
POKEY (Sound)	$E800	$D200
PIA	Not present	$D300
ANTIC	$D400	$D400

Most documentation seems to use the Atari 800 addresses... so we'll do the same... if you're using an Atari 5200 just change $D0 to $C0 for GTIA code, and $D2 to $E8 for POKEY code... of course, it's best to just define a symbol for each system in your ASM code - which is what our example ASM code will do!

Screen Modes

In this tutorial we'll be looking at two screen modes...

ANTIC mode F is a 4 color 160x192 pixel screen mode with 'Rectangular' pixels like Mode 0 on the CPC... the 4 colors are defined by 4 registers
COLBK (&D01A) sets the background... Colors 1-3 are set by COLPF0 ($D016), COLPF1 ($D017) and COLPF2 ($D018)

ANTIC mode E is a 2 color 320x192 pixel mode screen with roughly square pixels.. technically it only has one color - with two shades... COLBK (&D01A) sets the background brightness and color, and foreground color... and COLPF0 ($D016) sets the foreground brightness

Screen Display lists

The screen is defined in a similar way to the Elan Enterprise... we provide a Display List... which provides commands to the screen hardware... we can do clever tricks like have different screen modes for different areas of the screen if we

68000 Tutorial List
Learn 68000 Assembly
Hello World Series
Platform Specific Series
Grime 68000
68000 Downloads
68000 Cheatsheet
Sources.7z
DevTools kit
68000 Platforms
Amiga 500
Atari ST
Neo Geo
Sega Genesis / Mega Drive
Sinclair QL
X68000 (Sharp x68k)

8086 Content
Learn 8086 Assembly
Platform Specific Series
Hello World Series
8086 Downloads
8086 Cheatsheet
Sources.7z
DevTools kit
8086 Platforms
Wonderswan
MsDos

ARM Content
Learn ARM Assembly
Platform Specific Series
ARM Downloads
ARM Cheatsheet
Sources.7z
DevTools kit

want - but we won't be using them in this tutorial - we'll just have a simple list that has the screen all the same mode!...
Our Screen will start at &2000... unfortunately we do have to do some tricks, to allow our screen to 'step' over the 4k boundary at $3000 (otherwise it would wrap) - we do this by putting a new 'start memory position' command ($40+screen mode) - to move the memory position to $3000 - getting over the wrap limitation!

We also have to put 3 blank strips at the top, and a 'loop' in at the end of the screen, to allow normal working of the screen!

You'll need to set Smode to F or E for this example code to work!

```
 align 8
dlist   db      $70,$70,$70;$70 7= 8 blank lines 0= blank lines

        db $40+Smode,$60,$20        ;Strange start ($2060) to safely step over the boundary

        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode

        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode

        db $40+Smode,$00,$30        ;Have to manually step over the 4k boundary
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode

        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode
        db      Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode,Smode

        ;db 01
        db      $41         ;Loop
        dw       dlist
dlistEnd
```

Just creating a display list isn't enough, we need oto put it in the display list registers at $D402-$D403, and turn the DMA control on.

This will start the screen drawing with our chosen mode!

```
DMACTL  equ     $D400           ;DMA Control
DLISTL  equ     $D402           ;Display list lo
DLISTH  equ     $D403           ;Display list hi

        lda     #<dlist;$00     ;Set Display list pointer
        sta     DLISTL
        lda     #>dlist;$10
        sta     DLISTH

        lda     #%00100010      ;Enable DMA
        sta     DMACTL
```

The following byte commands can be used in a display list:

Command	Function
$x0-$xF	Screen mode change
$70	8 Blank lines
$4x $BB $AA	Start Screen mode x at address $AABB
$41 $BB $AA	Wait for Vblank, and restart display list at $AABB

ARM Platforms
Gameboy Advance
Nintendo DS
Risc Os

Risc-V Content
Learn Risc-V Assembly
Risc-V Downloads
Risc-V Cheatsheet
Sources.7z
DevTools kit

PDP-11 Content
Learn PDP-11 Assembly
PDP-11 Downloads
PDP-11 Cheatsheet
Sources.7z
DevTools kit

TMS9900 Content
Learn TMS9900 Assembly
TMS9900 Downloads
TMS9900 Cheatsheet
Sources.7z
DevTools kit
TMS9900 Platforms
Ti 99

6809 Content
Learn 6809 Assembly
6809 Downloads
6809/6309 Cheatsheet
Sources.7z
DevTools kit
6809 Platforms

Screen Modes

The Atari 800 & 5200 have a variety of modes - we can change mode every line of the screen - but some modes are 'taller' than others... here are the options - note: we're only going to look at modes E and F in these tutorials

Antic Mode	Basic Mode	Colors	Lines	Width	Bytes per Line	Screen Ram (Bytes)
2	0	2	8	40	40	960
3	N/A	2	10	40	40	760
4	N/A	4	8	40	40	960
5	N/A	4	16	40	40	480
6	1	5	8	20	20	480
7	2	5	16	20	20	240
8	3	4	8	40	10	240
9	4	2	4	80	10	480
A	5	4	4	80	20	960
B	6	2	2	160	20	1920
C	N/A	2	1	160	20	3840
D	7	4	2	160	40	3840
E	N/A	4	1	160	40	7680
F	8	2	1	320	40	7680

Atari 800 / 5200 Palette

xF	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF
x8	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
x0	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0

Atari 800 / 5200 Sprites

The Atari's hardware sprites are very weird!

Basically each sprite is 8 pixels wide and just 2 colors (1+transparent)... there are 4 'normal' ones that are 8 pixels wide.... and 4 missile sprites that are just 2(!) pixels wide... but we can position them together to give us 5 sprites.

Dragon 32/Tandy Coco
Fujitsu FM7
TRS-80 Coco 3
Vectrex

My Game projects
Chibi Aliens
Chibi Akumas

Work in Progress
Learn 65816 Assembly
Learn eZ80 Assembly

Misc bits
Ruby programming

Despite being 8 pixels wide... each sprite is up to 128 pixels tall (or 256 in hires mode) - the entire height of the screen!... if you can't guess this is because the systems is changing the data each rasterline.

The data used to draw the sprite is taken from a single pointer at **$D407**... if this pointer is set to $18 then all the sprites will use the $1800-$1FFF range - the exact address differs depending on whether the Resolution bit of $D400 is set to 0 or 1... in Res1 Sprites will be at $1800+$400 - $1C00 ... or in Res0 $1800+$200 = $1A00

On an Atari 800 where the GTIA is at $D000 this would give the following addresses for the sprite settings GTIA is at $C000 on the 5200)

The addresses controling the sprite are shown below... note we cannot set vertical position - we just write the sprite bitmap to a different address in the 'strip' of memory (eg between $1C00-$1CFF)

Player	Res0 Data	Res1Data	Width	Color	Xpos
0	$1A00+ypos	$1C00+ypos	$D008	$D012	$D000
1	$1A80+ypos	$1D00+ypos	$D009	$D013	$D001
2	$1B00+ypos	$1E00+ypos	$D00A	$D014	$D002
3	$1B80+ypos	$1F00+ypos	$D00B	$D015	$D003
4 (Missiles)	$1980	$1B00	$D00C	$D019* / $D012-$D015	$D004-$D007

*Missiles can be configured to use all 4 player colors for each 2 bit strip - or $D019 for all 4 2 bit strips ... this is set by PRIOR ($D01B)

The Sprites can be in front of, or behind the background... register **$D01B** (PRIOR) controls the order... and allows all 4 missiles to use color defined at $D019 as the sprite color - instead of the 4 player colors!
Note PRIOR is at **$D01B**... it seems to be incorrectly reported as $D10B or $D21B in some documentation!!!

To make use of sprites, we need to set the addresses shown above for the player sprite attributes, we also need to turn sprites on!

The example code to the right should do the job! note you needto set symbol GTIA to $D000 on the Atari 800, or $C000 on the Atari 5200

```
        lda #%00111110
        sta $D400           ;DMA control (SDMCTL)

        lda #$18 ;Sprites will be at $1800+$300 (or +$180 in low res mode)
        sta $D407    ;Store player sprite base

        lda #%00000011
        sta GTIA+$001D    ;Graphics Control (GRACTL)

        lda #%00010001   ;Priority: sprite 5 to use color 3
        sta GTIA+$1B       ;and put sprites in front of background
```

Atari 800 / 5200 Sprite Registers

Group	Name	Description	Address A80	Address A52	Bits	Notes
GTIA	HPOSP0	horizontal position of player 0	$D000	$C000		
GTIA	HPOSP1	horizontal position of player 1	$D001	$C001		
GTIA	HPOSP2	horizontal position of player 2	$D002	$C002		
GTIA	HPOSP3	horizontal position of player 3	$D003	$C003		
GTIA	HPOSM0	horizontal position of missile 0 (Player 4)	$D004	$C004		
GTIA	HPOSM1	horizontal position of missile 1 (Player 4)	$D005	$C005		
GTIA	HPOSM2	horizontal position of missile 2 (Player 4)	$D006	$C006		
GTIA	HOPSM3	horizontal position of missile 3 (Player 4)	$D007	$C007		
GTIA	SIZEP0	player 0 size	$D008	$C008	------WW	Width of sprite (0-3)
GTIA	SIZEP1	player 1 size	$D009	$C009	------WW	Width of sprite (0-3)
GTIA	SIZEP2	player 2 size	$D00A	$C00A	------WW	Width of sprite (0-3)
GTIA	SIZEP3	player 3 size	$D00B	$C00B	------WW	Width of sprite (0-3)
GTIA	SIZEM	missile size	$D00C	$C00C	wwWWwwWW	Width of sprite (Need to set all 4 parts)
GTIA	GRAFP0	player 0 graphics	$D00D	$C00D		(Used by DMA)
GTIA	GRAFP1	player 1 graphics	$D00E	$C00E		(Used by DMA)
GTIA	GRAFP2	player 2 graphics	$D00F	$C00F		(Used by DMA)
GTIA	GRAFP3	player 3 graphics	$D010	$C010		(Used by DMA)
GTIA	GRAFM	missile graphics	$D011	$C011		(Used by DMA)
GTIA	COLPM0	color/brightness, player/missile 0	$D012	$C012		
GTIA	COLPM1	color/brightness, player/missile 1	$D013	$C013		
GTIA	COLPM2	color/brightness, player/missile 2	$D014	$C014		

GTIA	COLPM3	color/brightness, player/missile 3	$D015	$C015		
GTIA	COLPF3	color/brightness of setcolor 3 / Player 5 (missile)	$D019	$C019		
GTIA	PRIOR	p/m priority and GTIA mode	$D01B	$C01B	GGmMpppp	G=gtia mode (0=normal) C=multiColor M=Missile (player 5) pppp=priority setting (1=sprites in front 4=behind)
GTIA	GRACTL	graphics control	$D01D	$C01D	------L45	Latch Trigger / Enable 4 player / enable 5 (missiles)
ANTIC	DMACTL	Direct Memory access control (DMA)	$D400	$C400		
ANTIC	PMBASE	player/missile address / 256	$D407	$C407		

Pokey Sound

Group	Name	Description	Address A80	Address A52	Bits	Meaning
POKEY	AUDF1	Audio frequency 1 control	$D200	$E800	FFFFFFFF	F=Frequency (0=highest tone)
POKEY	AUDC1	Audio channel 1 control	$D201	$E801	NNNNVVVV	N=Noise (0=noise / 10=Square wave) V=Volume (15=loudest)
POKEY	AUDF2	Audio frequency 2 control	$D202	$E802	FFFFFFFF	F=Frequency (0=highest tone)
POKEY	AUDC2	Audio channel 2 control	$D203	$E803	NNNNVVVV	N=Noise (0=noise / 10=Square wave) V=Volume (15=loudest)
POKEY	AUDF3	Audio frequency 3 control	$D204	$E804	FFFFFFFF	F=Frequency (0=highest tone)
POKEY	AUDC3	Audio channel 3 control	$D205	$E805	NNNNVVVV	N=Noise (0=noise / 10=Square wave) V=Volume (15=loudest)
POKEY	AUDF4	Audio frequency 4 control	$D206	$E806	FFFFFFFF	F=Frequency (0=highest tone)
POKEY	AUDC4	Audio channel 4 control	$D207	$E807	NNNNVVVV	N=Noise (0=noise / 10=Square wave) V=Volume (15=loudest)

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

POKEY	AUDCTL	general audio control	$D208	$E808	N1234HHS	N=Noise bit depth 1234=Channel Clocks HH=highpass filters S=main clockspeed

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)